НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут телекомунікаційних систем
Кафедра Інформаційно-телекомунікаційних мереж

«На правах рукопису»
УДК _____

«До захисту допущено»
Завідувач кафедри
_____ Лариса ГЛОБА
«___»_____2020 р.

Магістерська дисертація

на здобуття ступеня магістра
за освітньо-професійною програмою «Інформаційно-комунікаційні технології»
зі спеціальності 172 «Телекомунікації та радіотехніка»
на тему: «Удосконалений метод активації веб-сервісів на основі безсерверних хмарних обчислень»

Виконав:
студент ІІ курсу, групи ТІ-91мп
Черешня Віталій Радиславович _____

Керівник:
Старший викладач кафедри ІТМ ІТС, к.т.н.
Суліма Світлана Валеріївна _____

Консультант з 2 та 3 розділу:
Асистент кафедри ІТМ ІТС
Курдеча Василь Васильович _____

Рецензент:
Зав. кафедри промислової електроніки
КПІ ім. Ігоря Сікорського проф., д.т.н.
Ямненко Юлія Сергіївна _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут телекомунікаційних систем
Кафедра Інформаційно-телекомунікаційних мереж

Рівень вищої освіти – другий (магістерський)

Спеціальність – 172 «Телекомунікації та радіотехніка»

Освітньо-професійна програма «Інформаційно-комунікаційні технології»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Лариса ГЛОБА
«___»_____2020 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Черешні Віталію Радиславовичу

1. Тема дисертації «Удосконалений метод активації веб-сервісів на основі безсерверних хмарних обчислень», науковий керівник дисертації старший викладач кафедри інформаційно-телекомунікаційних мереж Суліма Світлана Валеріївна, к.т.н., затверджені наказом по університету від «03» листопада 2020 р. № 3208-с.

2. Термін подання студентом дисертації 11.12.2020 р.

3. Об’єкт дослідження: Використання веб-сервісів на основі безсерверних хмарних обчислень.

4. Предмет дослідження: Метод активації веб-сервісів на основі безсерверних хмарних обчислень.

5. Перелік завдань, які потрібно розробити:
   1. Провести аналіз проблем ефективності використання веб-сервісів на основі безсерверних хмарних обчислень та огляд існуючих методів їх вирішення.
   2. Проаналізувати існуючі методи покращення ефективності використання веб-сервісів на основі безсерверних хмарних обчислень та зформувати підхід до автоматичного обчислення коефіцієнта маштабування.

3. Вдосконалити метод активації веб-сервісів на основі безсерверних хмарних обчислень за рахунок автоматичного обчислення коефіцієнта маштабування.
4. Оцінити ефективність запропонованого рішення на основі натурного моделювання.
5. Створити стартап-проект.

6. Орієнтовний перелік ілюстративного матеріалу
   1. Тема, мета, актуальність, задачі дослідження.
   2. Аналіз існуючих методів підвищення ефективності використання веб-сервісів на основі безсерверних хмарних обчислень.
   3. Підхід до автоматичного обчислення коефіцієнта маштабування.
   4. Результати вимірювання часу відгуку до та після реалізації вдосконаленого методу.
   5. Опис запропонованого підходу як стартап-проекту.
   6. Загальні висновки.

7. Орієнтовний перелік публікацій

9. Дата видачі завдання 01.09.2019 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Вступ	20.09. – 29.09.2020	виконано
2	Аналіз проблеми ефективності використання	29.09 – 09.10.2020	виконано
3	Огляд існуючих методів вирішення проблеми	09.10 – 25.10.2020	виконано
4	Формування підходу до автоматичного обчислення коефіцієнта маштабування	25.10 – 31.10.2020	виконано
5	Моделювання тестової системи	31.10 – 07.11.2020	виконано
6	Розробка та тестування вдосконаленого методу	07.10 – 25.11.2020	виконано
7	Розробка стартап проекту	25.11 – 28.11.2020	виконано
8	Висновки	28.11 – 06.12.2020	виконано

Студент Віталій ЧЕРЕШНЯ

Науковий керівник дисертації Світлана СУЛІМА

РЕФЕРАТ

Робота містить 67 сторінок та 28 рисунків. Було використано 22 джерела.

Актуальність роботи: Метод активації дозволяє забезпечити наднизьку затримку у роботі з веб-сервісами на основі безсерверних хмарних обчислень. Ви передаєте програмі скільки екземплярів певного веб-сервісу ви хочете тримати активними і вона впорається з маштабуванням за вас. З часом трафік веб-сервісу може збільшитися, а це означає, що потрібно утримувати більше активних екземплярів, або менше, якщо навпаки. Потрібно мати деякий коефіцієнт маштабування, який буде автоматично обчислюватись беручи до уваги поточний трафік веб-сервіса. Це дозволить більш ефективно використовувати веб-сервіси на основі безсерверних хмарних обчислень та зменшити операційні затрати.

Об’єкт дослідження: Процес використання веб-сервісів на основі безсерверних хмарних обчислень.

Предмет дослідження: Метод активації веб-сервісів на основі безсерверних хмарних обчислень.

Мета роботи: Вдосконалити метод активації веб-сервісів на основі безсерверних хмарних обчислень для підвищення ефективності їх використання.

Для досягнення мети дослідження було поставлено та вирішено такі основні задачі:

1. Провести аналіз проблем ефективності використання веб-сервісів на основі безсерверних хмарних обчислень та огляд існуючих методів іх вирішення.
2. Проаналізувати існуючі методи покращення ефективності використання веб-сервісів на основі безсерверних хмарних обчислень та зформувати підхід до автоматичного обчислення коефіцієнта маштабування.

3. Вдосконалити метод активації веб-сервісів на основі безсерверних хмарних обчислень за рахунок автоматичного обчислення коефіцієнта маштабування.

4. Оцінити ефективність запропонованого рішення на основі натурного моделювання.

5. Розробити стартап-проект.

На основі проведеного аналізу виявлено проблему низької ефективності використання веб-сервісів на основі безсерверних хмарних обчислень. Аналіз показав, що одним із способів підвищення ефективності використання є вдосконалення методу активації веб-сервісів на основі безсерверних хмарних обчислень. Вдосконаливши метод активації, за рахунок автоматичного обчислення коефіцієнта маштабування активних екземплярів веб-сервіса вдалося підвищити ефективність використання цього веб-сервіса, що було підтверджено результатами проведеного аналізу. За результами роботи було розроблено стартап-проект для оцінки затрат на реалізацію методу активації та можливого прибутку від його вдосконалення.

Наукова новизна: Запропоновано підхід до автоматичного обчислення коефіцієнта маштабування який базується на вимірюванні трафіку веб-сервіса, що дозволить підвищити ефективність використання веб-сервісів на основі безсерверних хмарних обчислень.

Практична цінність: Результати дослідження забезпечують зменшення операційних витрат на впровадження методу активації веб-сервісів на основі безсерверних хмарних обчислень та більш ефективне викорастання хмарних ресурсів.

Методи дослідження: Основними методами дослідження є математичне та імітаційне моделювання.

Ключові слова: безсерверні обчислення, FaaS, трафік веб-сервіса, ефективність використання, час відгуку, холодний старт, метод активації, динамічне маштабування, асинхронні виклики, тестування навантаження, AWS Lambda.

ABSTRACT

The work contains 67 pages and 28 figures. 22 sources have been used.

Actuality: The activation method allows you to provide ultra-low latency in work with web services based on serverless cloud computing. You give the program how many instances of a particular web service you want to keep active and it will handle the scaling for you. Web service traffic may increase over time, which means that you need to keep more active instances, or less, if traffic decreases. You need to have some scaling factor that will be calculated automatically based on the current traffic of the web service. This will allow more efficient use of web services based on serverless cloud computing and reduce operating costs.

Object of research: The process of using web services based on serverless cloud computing.

Subject of research: Method of activating web services based on serverless cloud computing.

Aim of research: Improve the method of activating web services based on serverless cloud computing to increase their capacity utilization.

Research objectives:

1. Analyze the capacity utilization problems of web services based on serverless cloud computing and review the existing solutions.

2. Analyze existing methods for improving the capacity utilization of web services based on serverless cloud computing and develop an approach to automatic calculation of the scaling factor.

3. Improve the method of activating web services based on serverless cloud computing by automatically calculating the scaling factor.

4. Analyze the capacity utilization of web services of the test system before and after the improvement of the activation method.

5. Develop a startup project.

Based on the analysis, the problem of low capacity utilization of web services based on serverless cloud computing was revealed. The analysis showed

that one of the ways to increase the capacity utilization is to improve the method of activating web services based on serverless cloud computing. After improving the activation method, by automatically calculating the scaling factor of the active instances of the web service, it was possible to increase the capacity utilization of this web service, which was confirmed by the results of the analysis after implementation. Based on the results of the work, a startup project was developed to estimate the cost of implementation of the activation method and the possible benefits of its improvement.

Scientific novelty: Offered the approach to automatic calculation of scaling factor which is based on measurement of traffic of web service that will allow to increase the capacity utilization of web services based on serverless cloud computing.

Practical value: The results of the study provide a reduction in operating costs for the implementation of the method of activating web services based on serverless cloud computing and more efficient use of cloud resources.

Research methods: The main research methods are mathematical and simulation modeling.

Key words: serverless computing, FaaS, web service, capacity utilization, response time, cold start, activation method, auto-scaling, asynchronous requests, load testing, AWS Lambda.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API	Application Programming Interface
AWS	Amazon Web Services
EC2	Amazon Elastic Compute Cloud
ENI	Elastic Network Interface
FaaS	Function as a Service
S3	Amazon Simple Storage Service
SNS	Amazon Simple Notification Service
VM	Virtual Machine
VPC	Virtual Private Cloud

ВСТУП

Безсерверні обчислення є гарячою темою у світі архітектури програмного забезпечення. Під цією назвою розуміють програмне забезпечення, яке включає програми та служби, розміщені у хмарі, для керування логікою та станом сервера. Як правило, це різні веб-додатки, які використовують екосистему доступних у хмарі баз даних, служб аутентифікації тощо. Постачальники хмари пропонують обчислювальні середовища, відомі як FaaS-платформи, де запускається код таких програм. Назва «безсерверні» з’явилася тому, що рішення по управлінню сервером і плануванню потужностями веб-сервісів приховані від розробника або оператора. Це означає, що вам, не доведеться мати справу з сервером самостійно. Існують очевидні переваги таких програм у порівнянні з традиційними, але є і недоліки, такі як «холодний старт», які попри існуючі методи оптимізації все ще залишаються серйозними проблемами.

Метод активації дозволяє забезпечити паралельність веб-сервісів на основі безсерверних хмарних обчислень. Ви передаєте програмі скільки екземплярів певного веб-сервісу ви хочете тримати активними постійно і вона впорається з маштабуванням за вас. Це дозволяє використовувати веб-сервіси з наднизькою затримкою і без «холодного старту». З часом трафік веб-сервісу може збільшитися, а це означає, що потрібно утримувати більше активних екземплярів, або менше, якщо трафік зменшився. Потрібно мати деякий коефіцієнт маштабування, який буде автоматично обчислюватись беручи до уваги поточний трафік веб-сервіса. Це дозволить більш ефективно використовувати екземпляри веб-сервіса на основі безсерверних хмарних обчислень та зменшити операційні затрати. Робота виконана в рамках НДР кафедри №0119U001184: "Гетерогенна мережа збору, передачі та обробки інформації для системи розподіленої генерації MicroGrid." та №0116U005092 «Підвищення ефективності обробки даних зі споживчих пристроїв в телекомунікаційній мережі Інтернету Речей».

РОЗДІЛ 1
АНАЛІЗ ПРОБЛЕМ ЕФЕКТИВНОСТІ ВИКОРИСТАННЯ ВЕБ-СЕРВІСІВ НА ОСНОВІ БЕЗСЕРВЕРНИХ ХМАРНИХ ОБЧИСЛЕНЬ ТА ОГЛЯД ІСНУЮЧИХ МЕТОДІВ ІХ ВИРІШЕННЯ.

1.1. Аналіз проблем ефективності використання веб-сервісів на основі безсерверних хмарних обчислень

В загальному випадку, безсерверні обчислення означають програми, де серверна логіка все ще пишеться розробником, але, на відміну від традиційних систем, вони виконуються в контейнерах, які не зберігають свого стану після виклику, які викликаються подією (HTTP запит, запис в базі данних, повідомлення в черзі/каналі), ефемерні (можуть тривати лише в рамках одного виклику) і повністю керуються третьою стороною (хмарним провайдером). Через це їх ще називають «функціями». Модель виконання безсерверних обчислень виглядає наступним чином (рис. 1.1).

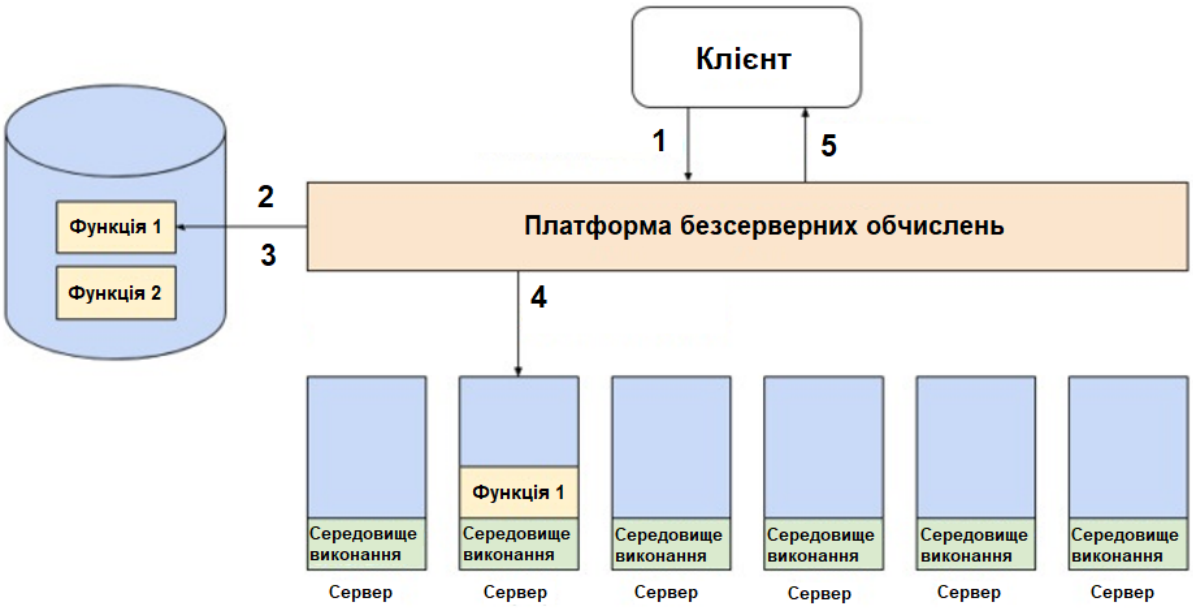

Рис. 1.1 Приклад загальної моделі виконання безсерверних обчислень

Пояснення до рисунку:

1. Клієнт робить запит на платформу безсерверних обчислень для виконання певної функції (сервісу).
2. Платформа безсерверних обчислень спочатку перевіряє, чи виконується функція на будь-якому з її серверів. Якщо функція ще не запущена, то платформа завантажує код цієї функції зі сховища.
3. Платформа потім розгортає функцію на одному з своїх серверів, які попередньо налаштовані з середовищем виконання, яке може запускати цю функцію.
4. Платформа виконує функцію і фіксує результат.
5. Платформа повертає результат назад клієнту.

Платформа безсерверних обчислень, така як AWS Lambda, дозволяє зберігати код ваших веб-сервісів і розгортати їх без необхідності конфігурування та керування базовими серверами. Вона ініціалізує нові екземпляри функцій тільки на вимогу (англ. on-demand). Щоразу, коли платформа отримує запит, але у неї немає активних екземплярів функції, вона призначає йому новий. Потім цей екземпляр функції повинен завантажити код функції або файл з кодом та відвантажити їх у пам’ять, перш ніж він зможе обробити запит. Процес ініціалізації коду вимагає часу, що значно збільшує затримку відповіді [3].

Коли ініціалізується код вашої функції, платформа проходить послідовність кроків, які спільно називаються «холодний старт»:

1. Платформа виділяє базовий ресурс VM для розміщення вашої функції.
2. Платформа створює контейнер Linux на виділеній VM, в якому буде запускатись ваш код.
3. Платформа прикріпляє ENI до контейнера, якщо ви вказали використовувати VPC.

4. Платформа копіює код до контейнера, який ви надали під час розгортання.
5. Платформа запускає середовище виконання, яке ви вказали, в межах контейнера.
6. Середовище виконання запускає код вашої функції.

Холодний старт необхідний, коли немає доступного контейнера для обробки події. Ця ситуація відбувається в наступні часи:

1. Коли змінюється код або конфігурація функції (в тому числі, коли розгортається перша версія функції).
2. Коли всі попередні контейнери були «утилізовані» через вік.
3. Коли всі попередні контейнери утилізувались через неактивність.
4. Коли функції потрібно масштабуватися, тому що всі поточні контейнери для необхідної функції вже обробляють події.

В той час як перші 2 випадки неминучі, останніх можна уникнути, змусивши функцію обробити фіктивні «пінг» події [1], розподіливши їх на потрібну кількість контейнерів фінкції за допомогою паралельних викликів цієї функції та повторюючи це періодично – цей метод носить назву «Метод активації веб-сервісів на основі безсерверних хмарних обчислень». Ідея методу полягає в тому, щоб мати подію в планувальнику, яка спрацьовує кожні N хвилин і надсилає M асинхронних запитів потрібній функції (рис. 1.2). Якщо всі ці запити потрапляють одночасно, платформа повинна забезпечити щонайменше M екземплярів функції для їх обробки. Фактична операція «активації» не повинна виконувати жодної корисної роботи, тобто бізнес-логіки, але вона скидає таймер утилізації назад до нуля.

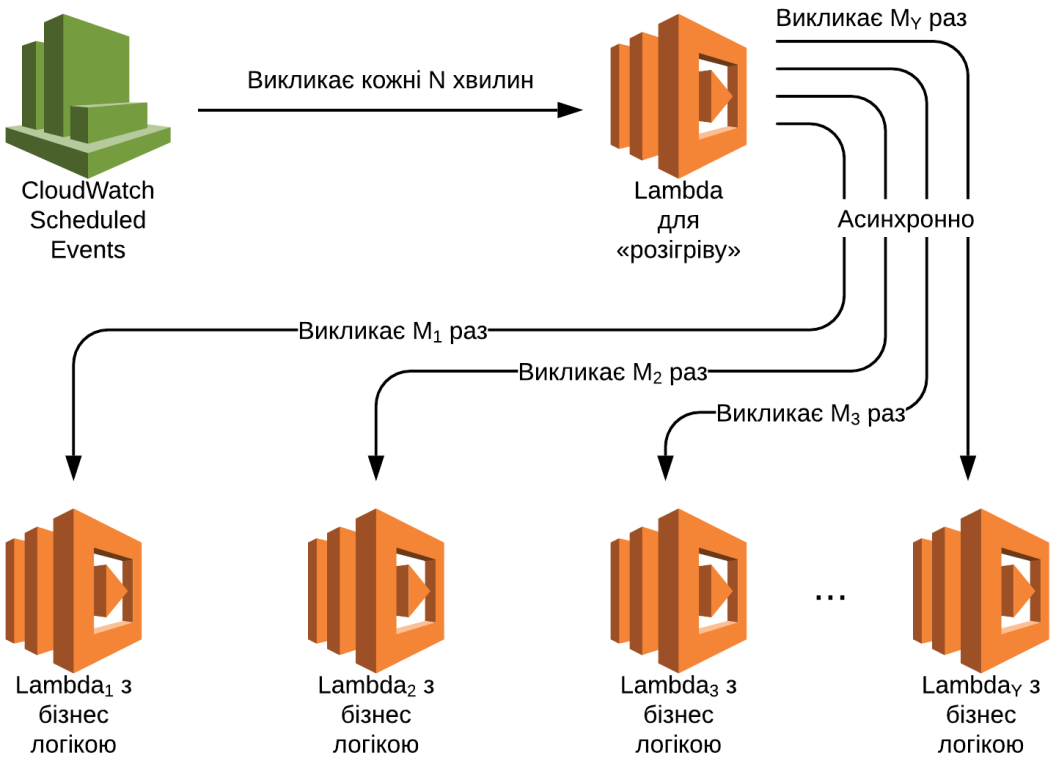

Рис. 1.2 Високорівнева архітектура методу активації веб-сервісів на прикладі AWS

Через деякий час навантаження на вашу функцію може збільшитися і як наслідок платформа почне створювати нові середовища виконання на вимогу щоб поглинути це навантаження, що призводе до збільшення часу відгуку за рахунок затримки «холодного старту» і зменшення ефективності цієї функції. Вихід – утримувати більше екземплярів данної функції постійно, щоб поглинути навантаження під часу піку. Вам також може знадобитися зменшити кількість екземплярів функції, якщо трафік слідує за шаблоном вниз, щоб звільнити ресурси та зменшити витрати. Для цього потрібно мати деякий коефіцієнт маштабування, який буде змінювати кількість асинхронних «пінг» запитів до функції автоматично в залежності від її трафіка. А щоб не обчислювати його вручну, можна вдосконалити метод активації таким чином, щоб коефіцієнт маштабування обчислювався автоматично, беручи до уваги поточне використання активних екземплярів данної функції.

Вдосконалення методу активації дозволить покращити ефективність використання веб-сервісів на основі безсерверних хмарних обчислень за рахунок автоматичного маштабування екземплярів цих веб-сервісів під час зміни трафіку, при цьому, тримаючи значення часу відгуку веб-сервіса низьким.

1.2. Огляд існуючих методів вирішення проблеми ефективності використання веб-сервісів на основі безсерверних хмарних обчислень

Відомо, що «холодний старт» трапляється, коли ви вперше викликаєте функцію, або коли функція викликається після тривалої неактивності. Вони також трапляються, коли платформа масштабує функцію, оскільки кожен новий екземпляр функції є новим середовищем виконання.

Раніше безсерверне товариство створювало програми для активації функцій [1], щоб підвищити ймовірність обробки запиту існуючим середовищем виконання. Це хороший підхід для розробки та тестування стабільних систем (рис. 1.3), або там, де вам не потрібна гіпер-готовність функцій.

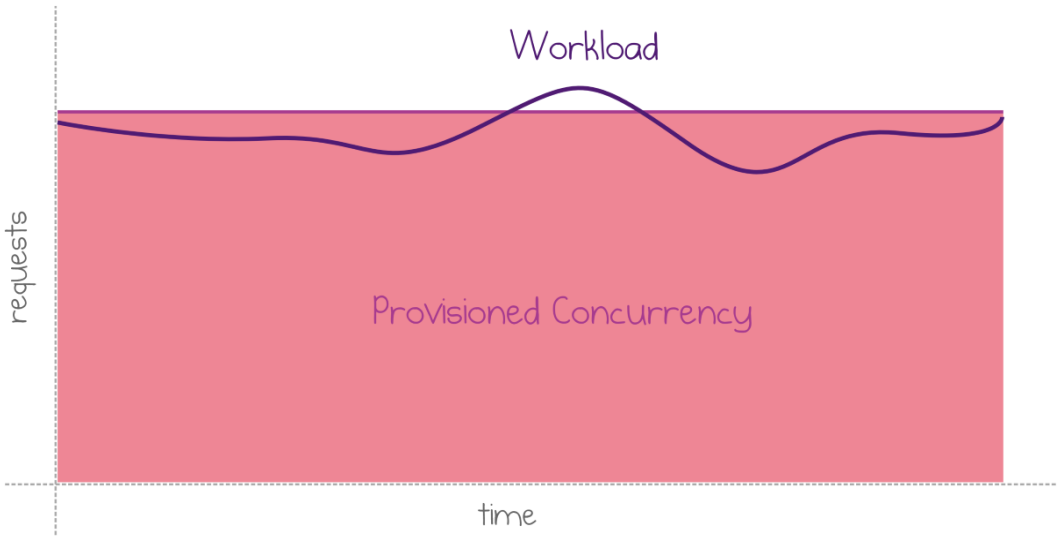

Рис. 1.3 Графік навантаження функції при застосуванні методу активації веб-сервісів на основі безсерверних хмарних обчислень

Однак для багатьох клієнт-орієнтованих систем навантаження сильно коливаються. Для нестабільних систем, які потребують низької затримки виконання функції, потрібно передбачати шаблон зміни трафіку цієї функції, щоб збільшити або зменшити кількість активних екземплярів відповідно.

Запланований профіль маштабування

Досить часто збільшення частоти запитів є частково передбачуваним. Наприклад, використання збільшується в робочий час і зменшується вночі, або це може бути будь-який інший проміжок часу в залежності від бізнесу. Збільшуючи кількість активних екземплярів функції в потрібний момент часу, як зображено на рис. 1.4, можна уникнути додаткової затримки через «холодний старт» та забезпечити кращий досвід для кінцевих користувачів. А завдяки можливостям планувальника, такого як Amazon CloudWatch Events, можна автоматизувати зміну кількості активних екземплярів функції на певну дату або час.

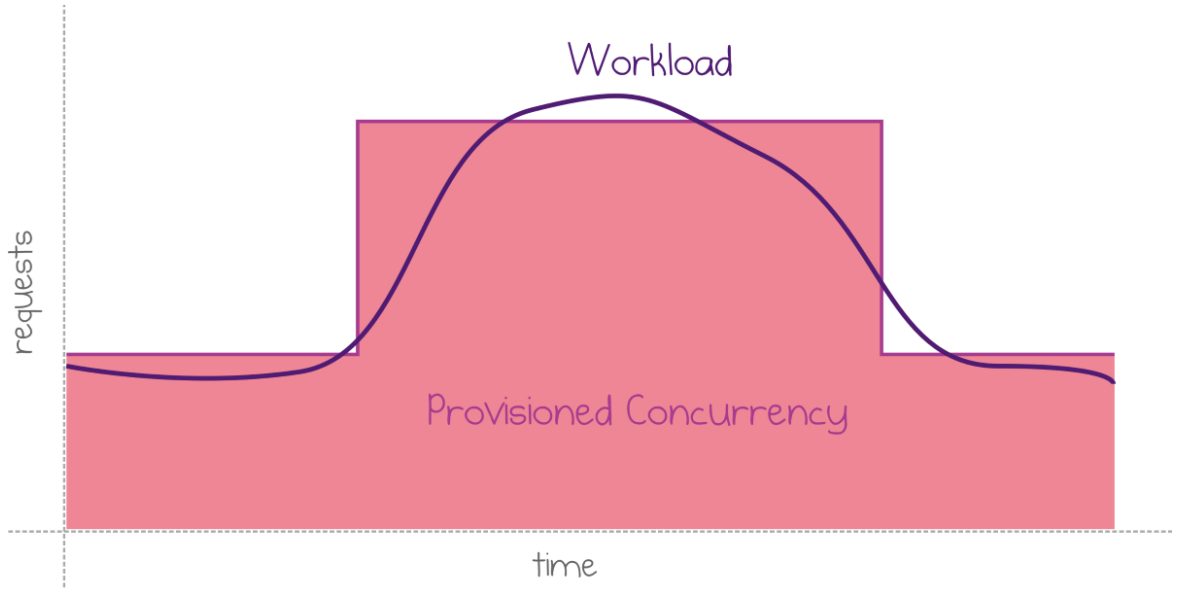

Рис. 1.4 Графік відповідності запланованого маштабування зміні навантаження функції

Динамічний профіль маштабування

Якщо шаблон робочого навантаження менш передбачуваний, потрібно налаштувати автоматичне маштабування активних екземплярів функцій на основі виміряного використання цієї функції, як показано на рис. 1.5. Цей метод передбачає спостережння за викликами функції, аналіз та збереження статистики у вигляді числових метрик. Після цього можна знаходити коефіцієнт маштабування, тобто збільшувати або зменшувати кількість активних екземплярів функціїї на основі метрики використання їх використання.

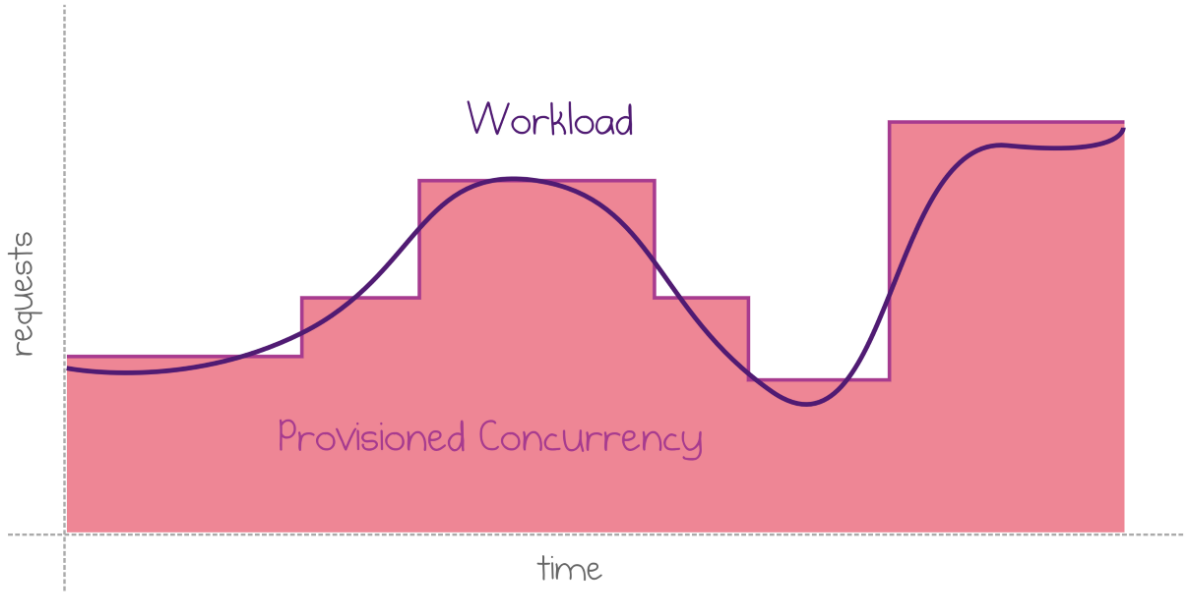

Рис. 1.5 Графік відповідності динамічного маштабування зміні навантаження функції

Недолік методу полягає у тому, що з’являються додаткові операційні витрати на спостереження, аналіз та збереження данних використання активних екземплярів функції необхідних для обчислення коефіцієнта маштабування, які можливо мінімізувати в залежності від реалізації.

Висновки

1. Проведено аналіз проблем ефективності використання веб-сервісів на основі безсерверних хмарних обчислень. Визначено, що головними проблемами є «холодний старт» та непередбачуваність трафіку реальних систем.
2. Розглянуто існуючі методи вирішення проблем ефективності використання веб-сервісів на основі безсерверних хмарних обчислень. Визначено, що для непередбачуваного трафіку підходить профіль динамічного маштабування активних екземплярів функції для якого потрібно аналізувати метрику використання, а для передбачуваного – підходить профіль запланованого маштабування.

РОЗДІЛ 2
АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ ДЛЯ ПОКРАЩЕННЯ ЕФЕКТИВНОСТІ ВИКОРИСТАННЯ ВЕБ-СЕРВІСІВ ТА ФОРМУВАННЯ ПІДХОДУ ДО АВТОМАТИЧНОГО ОБЧИСЛЕННЯ КОЕФІЦІЄНТА МАШТАБУВАННЯ

2.1. Аналіз існуючих методів покращення ефективності використання веб-сервісів на основі безсерверних хмарних обчислень

Маштабування активних екземплярів функцій дозволяє підготувати середовища виконання до отримання реального трафіку. Метод активації веб-сервісів на основі безсерверни хмарних обчислень окрім завантаження коду функції, також запускає код ініціалізації за межами основного обробника бізнес-логіки функції. Це забезпечує надійний спосіб підтримувати функції готовими протягом двозначної затримки в мілісекундах до можливості реагування на реальні запити від користувачів. Всі активні екземпляри функції (незалежно від середовища виконання) оброблюють запити швидше, ніж нові екземпляри, які були стоворені платформою на вимогу, щоб поглинути навантаження. Авжеш, такі середовища виконання як C# та Java, мають набагато повільніший час ініціалізації контейнера, ніж Node.js або Python, але швидший час виконання після ініціалізації. Якщо вдосконалити метод активації за рахунок підключення маштабування, ці середовища виконання виграють як від стабільно низької затримки через відсутність «холодних стартів», так і від продуктивності під час виконання.

Аналіз методу з використанням профілю запланованого маштабування

Збільшуючи кількість активних екземплярів функції в потрібний момент часу за розкладом можна уникнути додаткової затримки через

«холодний старт». Наприклад, розглянемо ситуацію, коли одна з компаній електронної комерції проводить акцію «угода дня» щодня опівдні. Під час неї в системі знижується ціна на конкретний товар на 60 хвилин. Функція, яка створює замовлення, назвем її CreateOrder, обробляє близько 20 запитів на секунду протягом дня. Опівдні зареєстрованим користувачам надсилається сповіщення, які потім підключаються до веб-сайту. Трафік різко зростає і може перевищувати 400 запитів за секунду для функції CreateOrder. Цей повторюваний шаблон показаний на рис. 2.1.

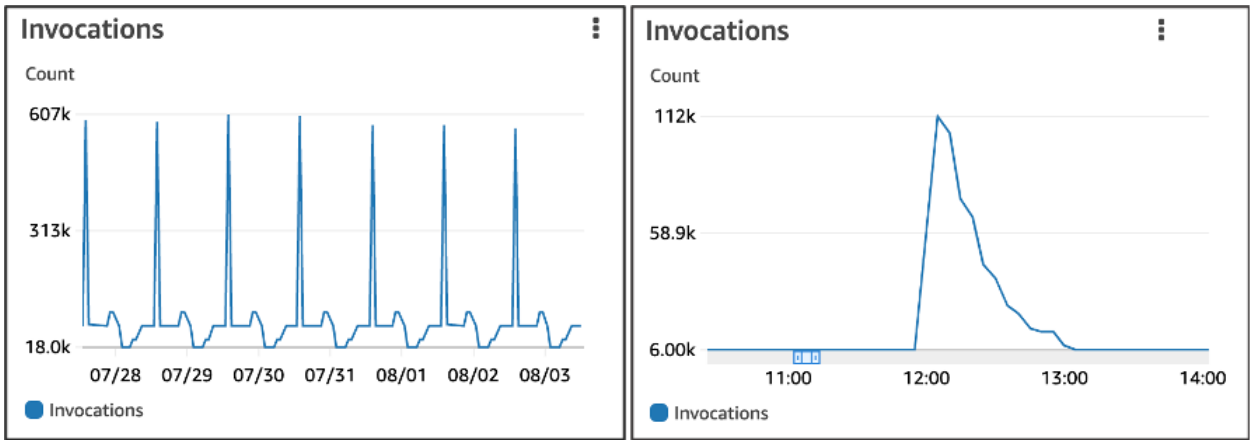

Рис. 2.1 Графік викликів Lambda функції CreateOrder в системі електронної комерції

Вивчаючи час відгуку функції в AWS X-Ray, перший графік показує нормальний трафік (рис. 2.2):

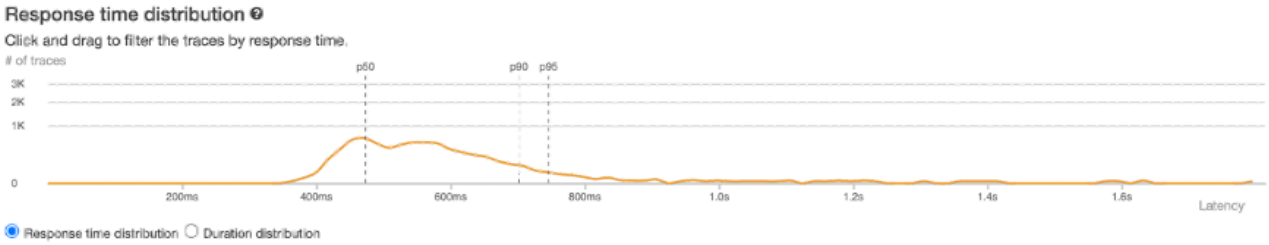

Рис. 2.2 Графік нормального розподілу часу відгуку Lambda функції CreateOrder в системі електронної комерції

Поки другий графік показує пік (рис. 2.3):

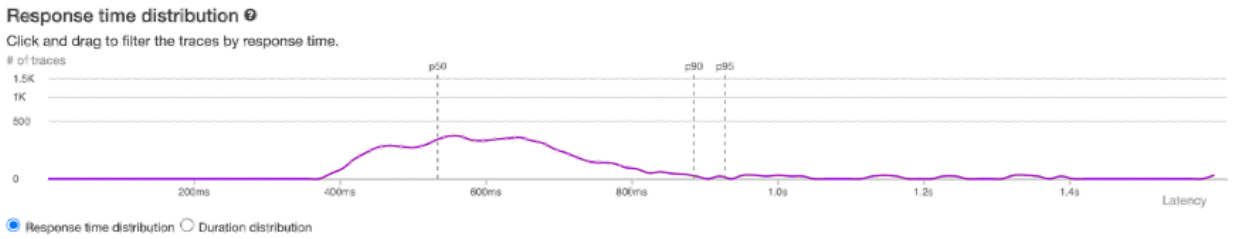

Рис. 2.3 Графік розподілу часу відгуку Lambda функції CreateOrder в системі електронної комерції під час піку

Середня затримка (p50) вища під час піку – 535 мс проти 475 мс при нормальному навантаженні. Значення p90 та p95 показують, що більшість викликів не перевищують 800 мс на першому графіку, але близько 900 мс на другому. Ця різниця обумовлена «холодним стартом», коли платформа AWS Lambda готує нові середовища виконання, щоб поглинути навантаження під час піку.

Інтегрувавши метод ативації веб-сервісів на основі безсерверних хмарних обчислень та збільшуючи кількість асинхронних «пінг» запитів опівдні можна уникнути цієї додаткової затримки та забезпечити кращий досвід для кінцевих користувачів. В ідеалі їх кількість також слід зменшити о 13:00, щоб уникнути зайвих витрат. А завдяки можливостям планувальника, такого як Amazon CloudWatch Events, можна налаштувати зміну кількісті асинхронних «пінг» запитів опівдні та о 13:00 на регулярній основі.

Необхідна кількість активних контейнерів функції, або ще паралельність функції, дорівнює середній кількості запитів за секунду, помноженій на середню тривалість функції. Наприклад, якщо середній час виконання функції – 500 мс, а робоче навантаження під час піку – 450 запитів на секунду, то потрібно регулярно надсилати 250 (450 * 0.5 + 10%) асинхронних «пінг» запитів, включаючи для 10% буферу, що має бути достатньо для поглинання заданого навантаження.

На наступному графіку затримки (рис 2.4) більшість запитів тепер виконуються менш ніж за 800 мс, відповідно до продуктивності протягом решти дня.

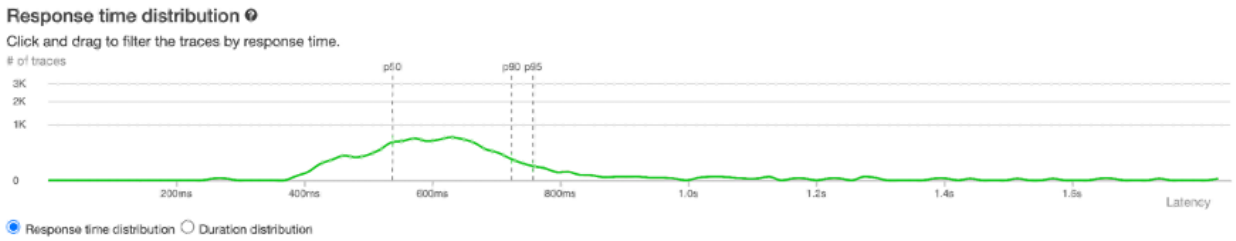

Рис. 2.4 Графік розподілу часу відгуку Lambda функції CreateOrder в системі електронної комерції під час піку після вдосконалення методу активації

Розподіл часу відгуку протягом «угоди дня» відтепер порівнянний з будь-яким іншим часом. Це допомагає забезпечити стабільну взаємодію з користувачами навіть під час великих навантажень. А налаштувавши метод активації веб-сервісів на основі безсерверних хмарних обчислень збільшувати кількість асинхронних «пінг» запитів на початок акції об 11:45 та зупинку о 13:15 кожного дня також допоможе оптимізувати витрати, обмежуючи використання профілю маштабування до 90 хвилин на день.

Аналіз методу з використанням профілю динамічного маштабування

Ми також можемо використовувати профіль динамічного масштабування, щоб автоматизувати надання відповідної потужності під час зміни трафіку. Безсерверні платформи, такі як AWS Lambda, підтримують автоматичне маштабування «із коробки», але кожен раз буде створюватись нове середовище виконання на вимогу, що призводить до збільшення затримки за рахунок «холодного старту». Для вирішення проблеми «холодного старту» ми вже знаємо, що потрібно застосувати метод активації веб-сервісів на основі безсерверних хмарних обчислень. Але змусити динамічне маштабування працювати в парі з методом активації – не така тривіальна задача. Для того щоб розібратись, пропоную розглянути ці два способи окремо та провести необхідні вимірювання, щоб після вдосконалення методу активації ми могли порівняти результати. На цей раз я буду використовувати веб-додаток Ask Around Me як приклад.

У веб-додатку Ask Around Me користувачі задають питання та відповідають на них у своєму географічному регіоні. Очікуване погодинне завантаження – 1000 нових запитань, 10 000 нових відповідей та 50 000 запитів на пошук запитань. Я використовуватиму ці цифри як основу для тестів. Для простоти, я буду перевіряти лише частину веб-додатку Ask Around Me, архітектура якої зображена на рис. 2.5.

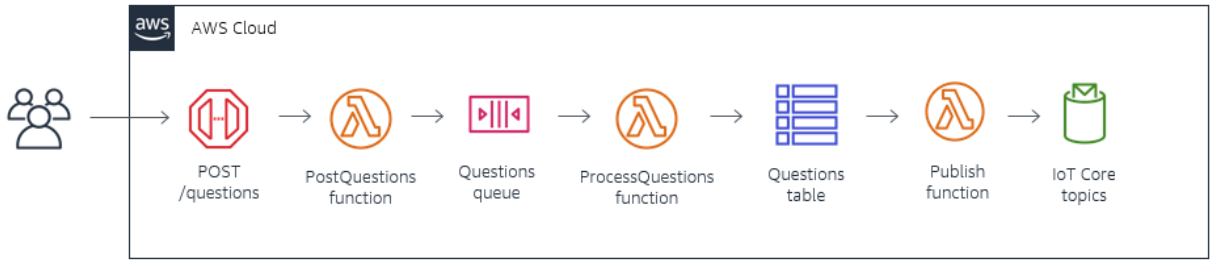

Рис. 2.5 Високорівнева архітектура веб-додатку Ask Around Me

Я запрограмував створення випадкової кількості POST /questions API запитів користувачів від 3 до 20 кожну секунду. Мінімально це відповідає приблизно 10800 запитам коричтувачів протягом 1 години, що вже перевищує передбачуване навантаження для цієї частини веб-додатку. Після тестування я отримав наступні результати (рис. 2.6).

```
All virtual users finished
Summary report @ 12:27:12(+0000) 2020-05-15
  Scenarios launched:  600
  Scenarios completed: 600
  Requests completed:  600
  Mean response/sec: 19.8
  Response time (msec):
    min: 136.3
    max: 2149.3
    median: 174.6
    p95: 1043.8
    p99: 1699.1
  Scenario counts:
    0: 600 (100%)
  Codes:
    200: 600
```

Рис. 2.6 Результати тестування навантаження ділянки веб-додатку Ask Around Me

У тесті навантаження протягом 5 хвилин медіана часу відгуку становить 175 мс, а найповільніше значення - 2149 мс. Медіана часу відгуку,

ймовірно, є прийнятною для користувача, тоді як будь-який час відгуку повільніший за 2 секунди робить взаємодію з додатком вже не настільки комфортним.

Я повторно запустив тест навантаження для цієї ділянки, щоб зібрати більш інформативні данні за допомогою AWS X-Ray (рис. 2.7).

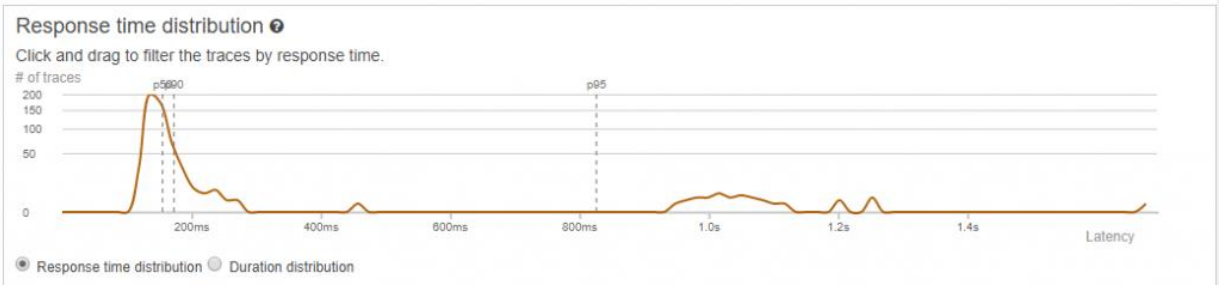

Рис. 2.7 Результати тестування навантаження ділянки веб-додатку Ask Around Me у AWS X-Ray

Я вибрав всі виклики функції PostQuestions на графіку після маркера p95, представляючи найповільніші 5% від усіх запитів. У результаті воно відфільтрувало 34 запити, які відповідають кількості активних екземплярів функції, які було створено платформою на вимогу при автоматичному маштабуванні (рис. 2.8).

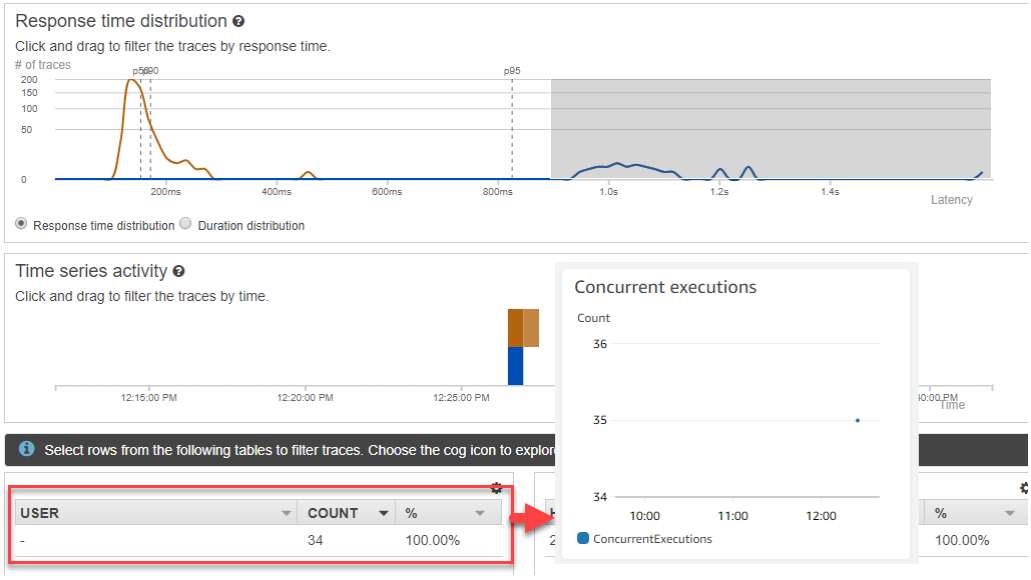

Рис. 2.8 Результати тестування навантаження ділянки веб-додатку Ask Around Me у AWS X-Ray

Вибравши один такий запит можна побачити тривалість кожного сегмента процесу (рис. 2.9).

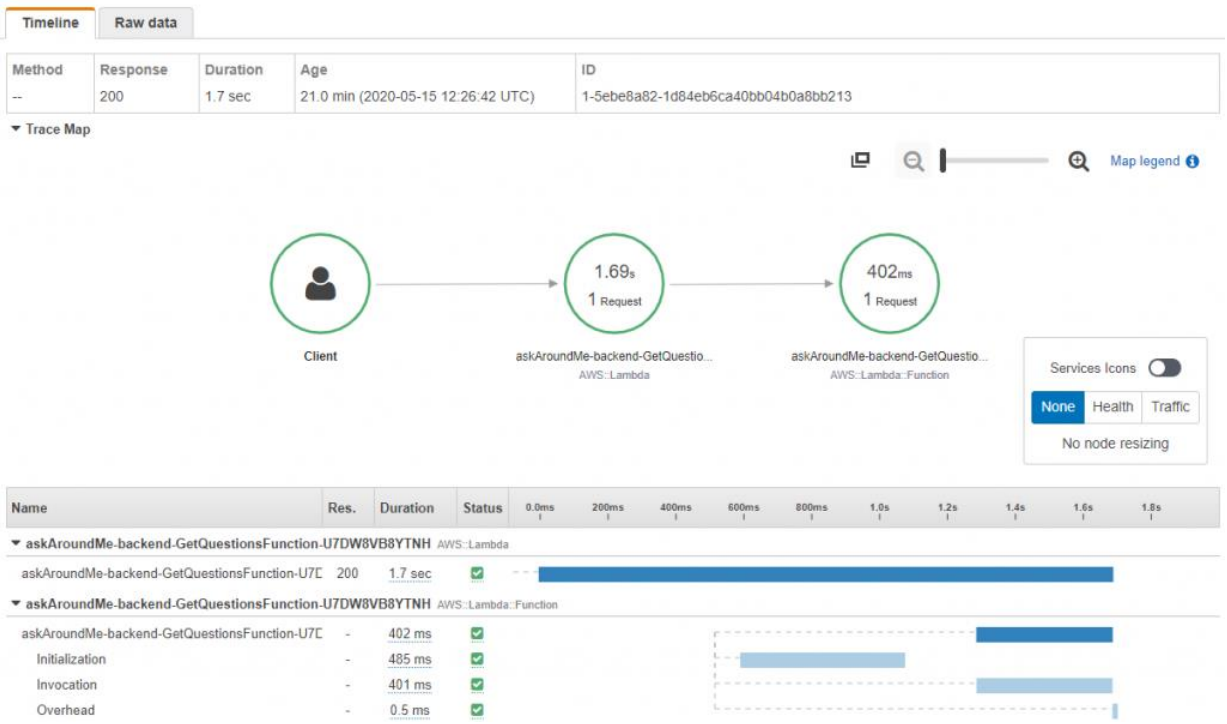

Рис. 2.9 Результати тестування навантаження ділянки веб-додатку Ask Around Me у AWS X-Ray

Цей аналіз показує, що на роботу функції впливає «холодний старт», тому що ініціалізація середовища виконання та коду функції займає більше 1 секунди.

Увімкнувши метод активації для цієї функції, можна значно зменшити час відгук [1] для 95% викликів функції, тобто ми максимально приблизити значення маркера p95 до медіани, яка приблизно дорівнює 175 мс та є прийнятною для користувача, а отже покращити ефективність функції.

Замість того, щоб резервувати фіксовану кількість активних екземплярів функції, можна налаштувати профіль динамічного маштабування, який буде збільшувати кількість асинхронних «пінг» викликів для методу активації під час росту трафіка функції і зменшуватиме – в зворотньому випадку, тим самим покращуючи ефективність використання екземплярів цієї функції не жертвуючи при цьому часом відгуку.

Потрібно розробити підхід до автоматичного обчислення коефіцієнта маштабування активних екземплярів функції та вдосконалити існуючий метод активації веб-сервісів на основі безсерверних хмарних обчислень.

2.2. Формування підходу до автоматичного обчислення коефіцієнта маштабування

Метод активації ґрунтується на асинхронному надсиланні програмою M фіктивних «пінг» запитів до функції яку потрібно активувати, тобто мати M активних екземплярів цієї функції. Через деякий час бездіяльності функції FaaS платформа почне утилізувати активні екземпляри, щоб звільнити хмарні ресурси. Щоб постійно тримати екземпляри функції активними потрібно регулярно надсилати M асинхронних запитів до неї. Значення M та унікальні ідентифікатори функцій передаються на вхід програми, яка періодично запускається засобами планувальника. Значення M задається вручну та дорівнює середньому часу виконання функції, помноженому на робоче навантаження функції під час піку, плюс деякий залишок на буфер. Щоб автоматично обчислювати M потрібно також брати до уваги поточний трафік функції яка розглядається, що є досить не тривіальним завданням та вимагає значних операційних ресурсів. Скориставшись власноруч створеною метрикою використання активних екземплярів функції ми можемо робити висновки щодо поточного трафіка цієї функції вцілому [2].

Наприклад, ми хочемо тримати використання активних екземплярів функції U близько 70%. Коли поточне використання екземплярів функції U' стабільно перевищує 77% (110% від запланованих 70%) що відповідає зміні трафіка функції за шаблоном вгору, ми можемо збільшити значення M на коефіцієнт маштабування X, де

$$X = \left(\frac{U' \times M}{U} - M\right)$$

для активації більшої кількості екземплярів, а коли поточне використання екземплярів функції U' стабільно менше 63% (90% від запланованих 70%) що відповідає зміні трафіка функції за шаблоном вниз, то ми можемо зменшити значення M на коефіцієнт маштабування X, де

$$X = \left(M - \frac{U' \times M}{U}\right)$$

В результаті нове значення кількості активних екземплярів функції M' буде приблизно дорівнювати $M + X$ або $M - X$ відповідно, а кількість активних екземплярів функції, яка раніше відповідала U' (≥ 77% або ≤ 63%) тепер буде приблизно відповідати заданому U, тобто 70%.

Реалізація подібної метрики використання активних екземплярів функції в значній мірі залежить від вибраної FaaS платформи, архітектури системи (які хмарні ресурси вам доступні) та середовища виконання функції. Визначившись з місцем зберігання значень метрики та алгоритмом їх оновлення, ми зможемо діставати поточне значення метрики та використовувати його для подальшого обчислення коефіцієнта маштабування.

Висновки

1. Проаналізовано різні методи підвищення ефективності використання веб-сервісів на основі безсерверних хмарних обчислень. На основі проведеного аналізу було обрано профіль динамічного маштабування для вдосконалення методу активації веб-сервісів на основі безсерверних хмарних обчислень, тому що він більше підходить для реальних систем у яких трафік непередбачуваний.
2. Запропоновано підхід до автоматичного обчислення коефіцієнта маштабування. Визначено, що створивши метрику використання активних екземплярів функції ми можемо робити висновки щодо поточного трафіка цієї функції та збільшувати або зменшувати значення кількості асинхронних «пінг» викликів для методу активації M на коефіцієнт маштабування X.

РОЗДІЛ 3
ВДОСКОНАЛЕННЯ МЕТОДУ АКТИВАЦІЇ ВЕБ-СЕРВІСУ НА ОСНОВІ БЕЗСЕРВЕРНИХ ХМАРНИХ ОБЧИСЛЕНЬ ЗА РАХУНОК АВТОМАТИЧНОГО ОБЧИСЛЕННЯ КОЕФІЦІЄНТА МАШТАБУВАННЯ

3.1. Архітектура тестової системи

У цьому розділі буде показано, як усунути затримку «холодного старту» в безсерверних архітектурах, що підтримують реальні веб-додатки з непередбачуваним трафіком, при цьому, покращивши використання їх екземплярів за рахунок динамічного маштабування. Як приклад системи, я буду використовувати веб-додаток Ask Around Me розміщенний в інфраструктурі Amazon, який вже згадувався раніше. Ця система дозволяє користувачам задавати та відповідати на запитання у своєму географічному регіоні. Веб-додаток використовує архітектуру яка зображена на рис. 3.1.

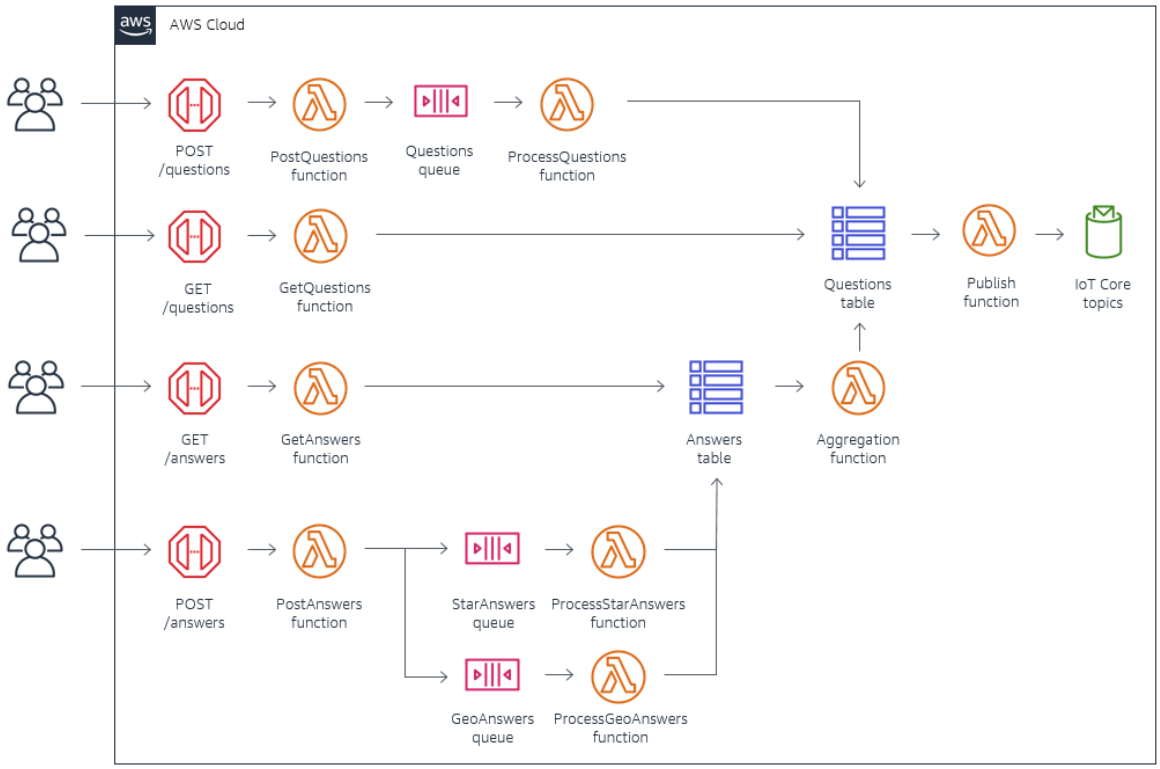

Рис. 3.1 Високорівнева архітектура веб-додатку Ask Around Me

Як і раніше, я буду перевіряти лише частину веб-додатку Ask Around Me, яка відповідає за створення запитань та архітектура якої зображена на рис. 3.2.

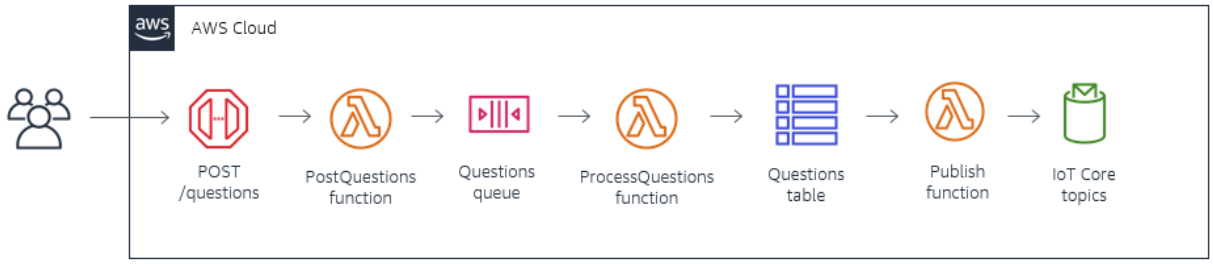

Рис. 3.2 Високорівнева архітектура ділянки веб-додатку Ask Around Me

Так як нас цікавлять лише функції які впливають на час відгуку веб-додатку, то нам потрібно «активувати» Lambda функцію PostQuestions, та ж сама функція що фігурувала у попередньому розділі під час аналізу.

Тестувати нашу ділянку системи ми будемо за допомогою API шлюзу POST /questions, який передаватиме виклики Lambda функції, та Artillery Community Edition – інструмента з відкритим кодом для тестування безсерверних API. Потрібно лише налаштувати кількість запитів на секунду та загальну тривалість тесту, і програма зробить все за нас, використовуючи «безголовий» (англ. headless) браузер Chromium для запуску своїх тестових потоків.

За збір данних часу відгуку та створення діаграм відповідатимуть служби Amazon CloudWatch Metrics та AWS X-Ray, результат роботи яких ми вже бачили у попередньому розділі.

3.2. Інтегрування методу активації веб-сервісів до тестової системи та його вдосконалення

Ми вже знаємо екземпляри якої функції потрібно активувати, залишилось вирішити «як саме» та «як часто» ми це будемо робити. В бакалаврській роботі [1] я активував функції кожні 5 хвилин (рис. 3.3), щоб не дати платформі безсерверних обчислень утилізувати їх екземпляри, що

було цілком достатньо. Фіксоване значення асинхронних «пінг» викликів M зберігалося в конфігурації середовища Warming Lambda функції.

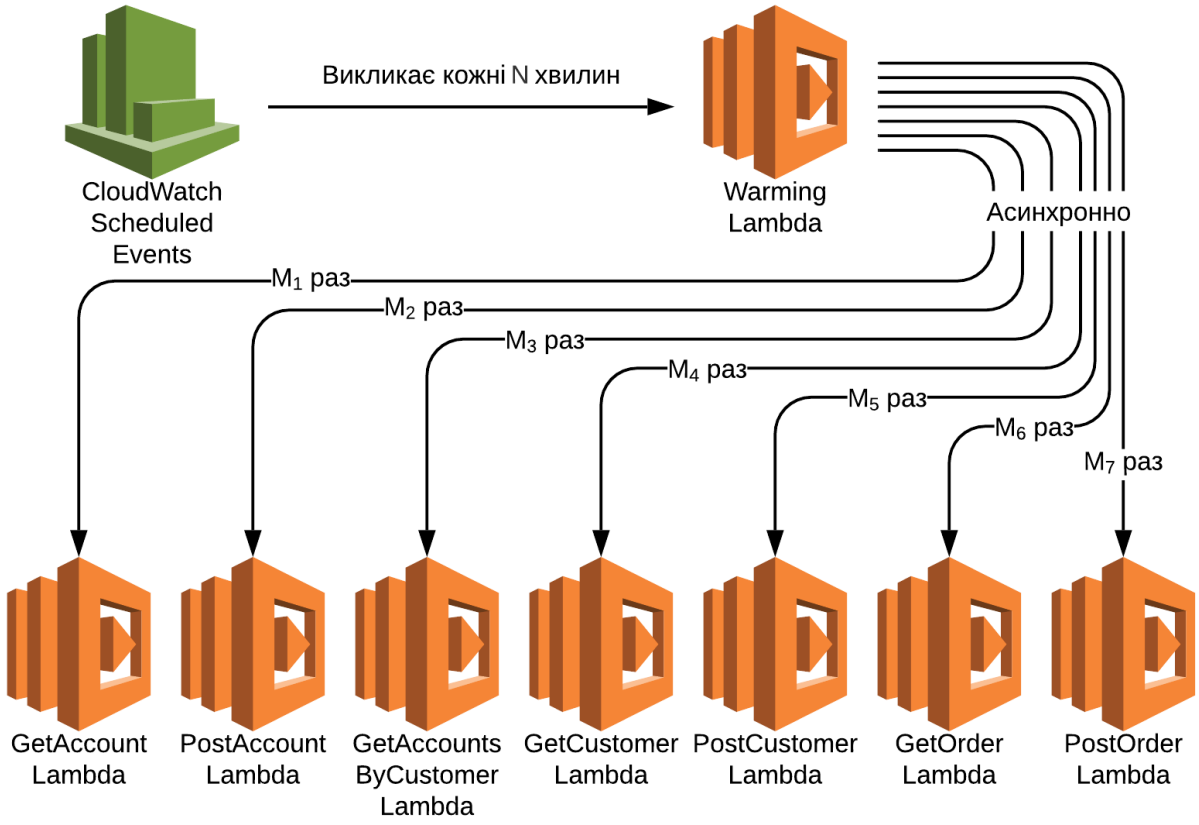

Рис. 3.3 Високорівнева архітектура методу активації тестової системи на прикладі бакалаврської роботи

Так само як і в бакалаврській роботі, в тестовій системі Lambda функція для «розігріву» робитиме M асинхронних «пінг» викликів до Lambda функції PostQuestions (рис. 3.4), яка має відрізнити фіктивний виклик від справжнього запиту користувача та зачекати менше секунди щоб розподілити запити на M окремих екземплярів функції PostQuestions.

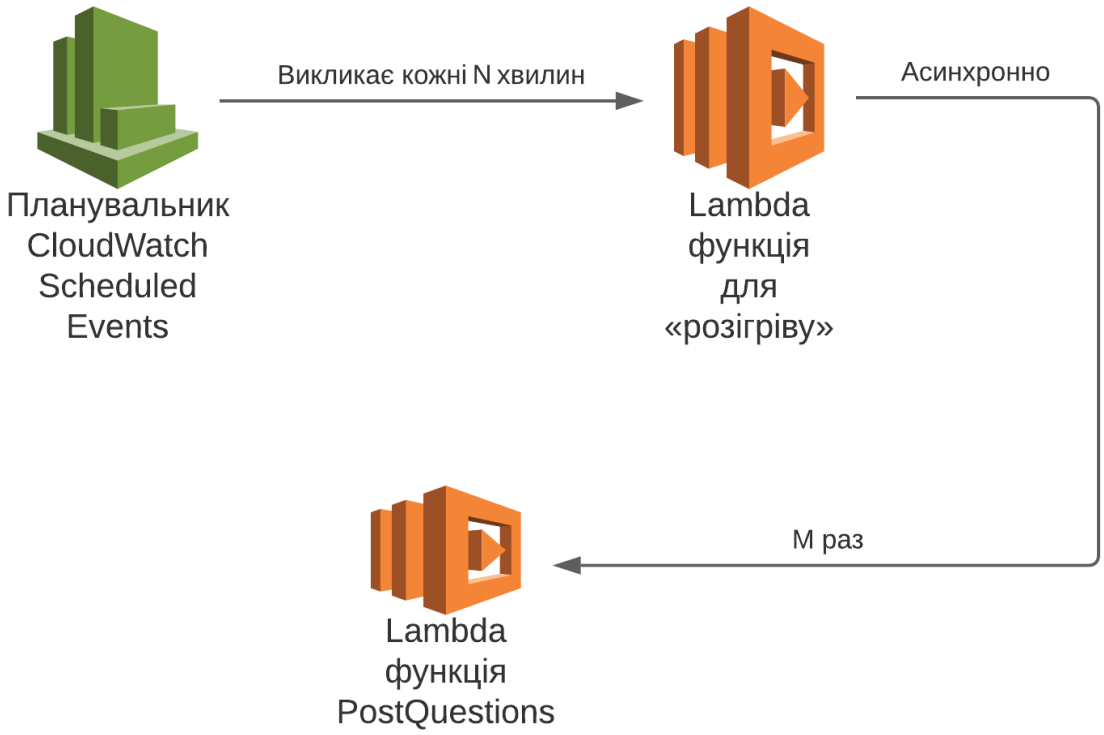

Рис. 3.4 Високорівнева архітектура методу активації Lambda функції PostQuestions тестової системи

Вдосконалення методу активації

На цей раз значення асинхронних «пінг» викликів M не буде зберігатись в конфігурації середовища Lambda функції для «розігріву». Щоб мати змогу змінювати значення M без зміни конфігурації функції для «розігріву» (адже при зміні конфігурації, змінюється версія функції та створюється новий контейнер) ми будемо передавати його в тілі запиту, але вже не від планувальника CloudWatch Scheduled Events, а іншої Lambda функції для «маштабування», яка це значення буде обраховувати. А викликаючи Lambda функцію для «маштабування» за розкладом кожні N хвилин ми збережемо цілісність методу активації функції PostQuestions тестової системи, як це зображено на рис. 3.5.

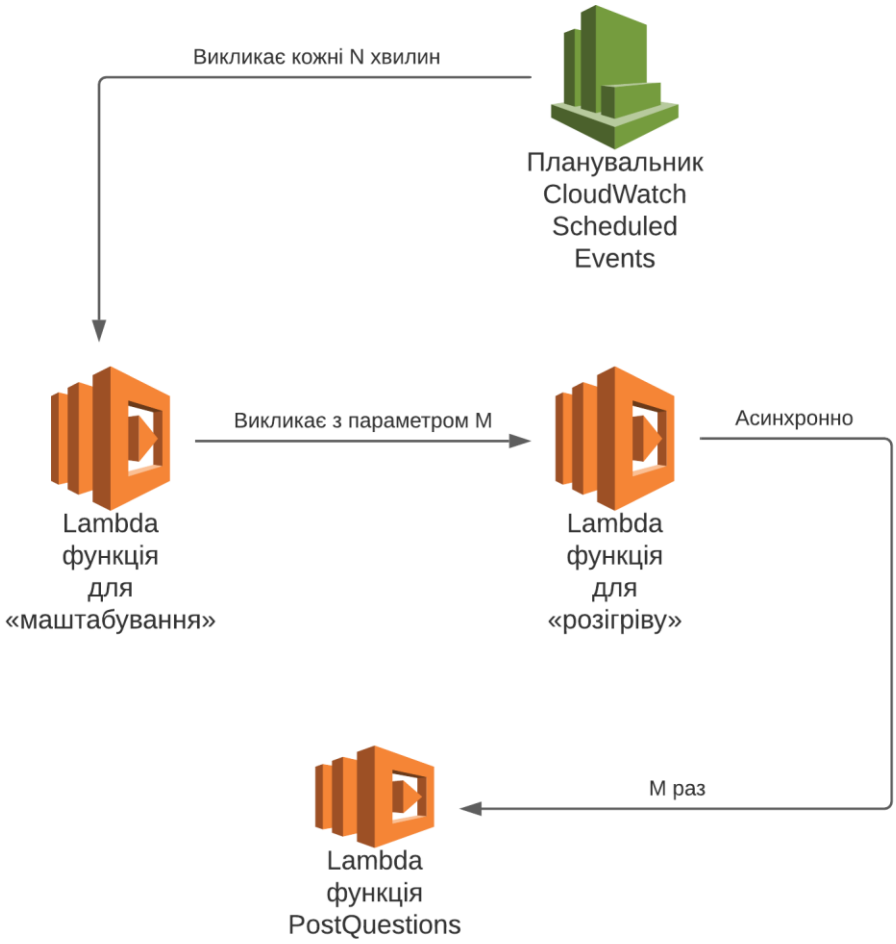

Рис. 3.5 Високорівнева архітектура вдосконаленого методу активації Lambda функції PostQuestions тестової системи

Залишається тільки реалізувати механізм за допомогою якого значення M буде автоматично обчислюватись беручи до уваги метрику використання активних екземплярів функцій тестової системи.

3.3. Реалізація механізму автоматичного обчислення коефіцієнта маштабування активних екземплярів функцій тестової системи

Ми вже знаємо, щоб запобігти утилізації екземплярів функції, потрібно надсилати «фіктивні» запити з певною частотою. Звичайно, потрібно внести необхідні зміни в нашу функцію PostQuestions тестової системи, щоб розрізняти «пінг» події та справжні запити клієнтів [1]. Завдяки запитам на «розігрів», платформа безсерверних обчислень буде тримати контейнери активними, але не все так просто.

Так як ми використовуємо функцію PostQuestions у виробництві (для справжніх клієнтів), потрібно зробити M асинхронних запитів на «розігрів», щоб зберегти M активних контейнерів. На цьому етапі існують два ризики:

– Ви тримаєте всі ваші контейнери зайнятими обробкою «фіктивних» подій, і справжній запит клієнта не може знайти місце для виконання. Це спричинить «холодний старт» для справжнього виклику.
– Функція в одному контейнері може затримати всі виклики одночасно, якщо вони досить швидко обробляються, і це може зробити інші контейнери неактивними через деякий час.

Щоб вирішити ці проблеми, ми повинні трохи зачекати на стороні функції, тобто призупинити головний потік функції на деякий час. Таким чином, один екземпляр функції не зловить усі «пінг» запити. А змусивши головний потік очікувати не більше 150 мс, ми переконаємося що блокуємо тільки шосту частину викликів за секунду.

Lambda функція для «розігріву»

Розглянемо більш детально реалізацію Lambda функції для «розігріву» нашої тестової системи. Як ми вже знаємо вона має вміти робити наступне:

1. Приймати запит від Lambda функції для «маштабування» з параметром M в тілі запита.
2. Асинхронно викликати задані Lambda функції передаючи їм на вхід «фіктивну» подію.
3. Повідомляти про результат операції.

Щоб задовільнити цим умовам наша Lambda функція повинна:

1. Прийняти на вхід об’єкт тіла запиту.
2. Зчитати данні з об’єкта тіла запиту, які містять список із ідентифікаторів функцій та кількості контейнерів M для них.
3. Виконати процесс «розігріву» на основі цих данних.

4. Якщо операція пройшла не успішно, то залогувати повідомлення про помилку.

Блок-схема алгоритму зображена на рис. 3.6.

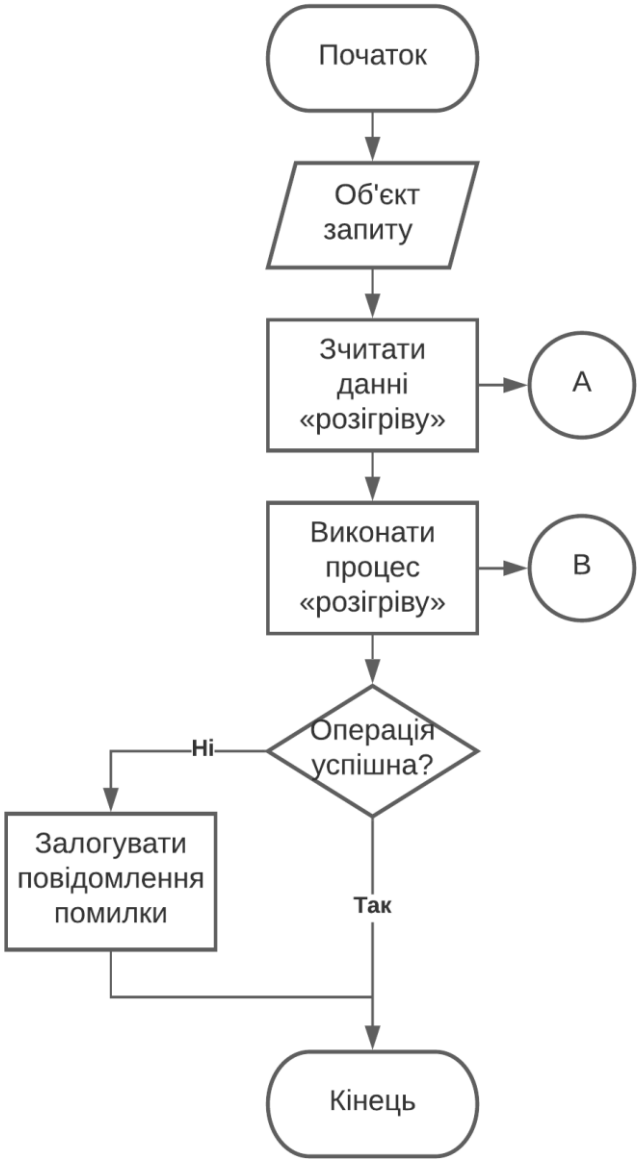

Рис. 3.6 Блок-схема алгоритму програми Lambda функції для «розігріву» тестової системи

Блок-схема алгоритму процеса зчитування данних «розігріву» зображена на рис. 3.7.

Рис. 3.7 Блок-схема алгоритму процеса зчитування данних «розігріву»

Lambda функції для «розігріву» тестової системи

Блок-схема алгоритму процесу «розігріву» зображена на рис. 3.8.

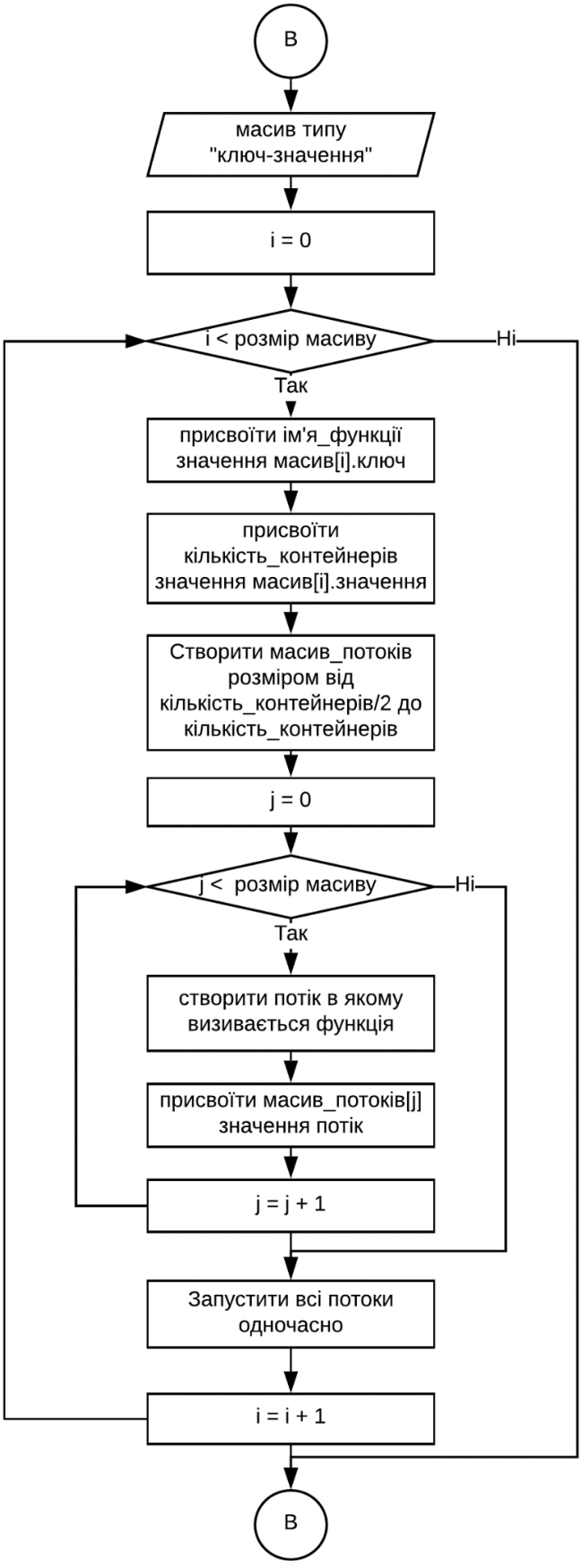

Рис. 3.8 Блок-схема алгоритму процеса «розігріву» Lambda функції для «розігріву» тестової системи

Так як потоки будуть виконуватися одночасно то в кінці достатньо буде лише залогувати результат операції. В результаті, операція «розігріву» завершиться тоді коли закінчить своє виконання останній потік.

Lambda функція для «маштабування»

Розглянемо більш детально реалізацію Lambda функції для «маштабування» нашої тестової системи. Вона має вміти робити наступне:

1. Приймати запит від планувальника CloudWatch Scheduled Events.
2. Обчислювати значення метрики використання активних екземплярів за останні N хвилин для кожної функції.
3. Обчислювати значення коефіцієнта маштабування X та значення кількості контейнерів M для кожної функції.
4. Зберігати нове значення кількості контейнерів M для кожної функції.
5. Викликати Lambda функцію для «розігріву».

Щоб задовільнити цим умовам наша Lambda функція повинна:

1. Прийняти на вхід об’єкт тіла запиту.
2. Зчитати данні з об’єкта тіла запиту, які містять список із ідентифікаторів функцій та початкової кількості контейнерів M для кожної з них.
3. Дістати збережені значення кількості контейнерів M для кожної функції з CloudWatch Metrics. Якщо їх немає, то використовувати початкові значення кількості контейнерів M.
4. Обчислити значення метрики використання активних екземплярів кожної функції за останні N хвилин, коефіцієнт маштабування та нове значення кількості контейнерів M, при цьому, зберігши його як метрику для кожної функції в CloudWatch Metrics та перезаписавши старе значення в об’єкті тіла запиту.
5. Викликати функцію для «маштабування» передавши їй на вхід об’єкт тіла запиту.

6. Якщо операція пройшла не успішно, то залогувати повідомлення про помилку.

Блок-схема алгоритму зображена на рис. 3.9

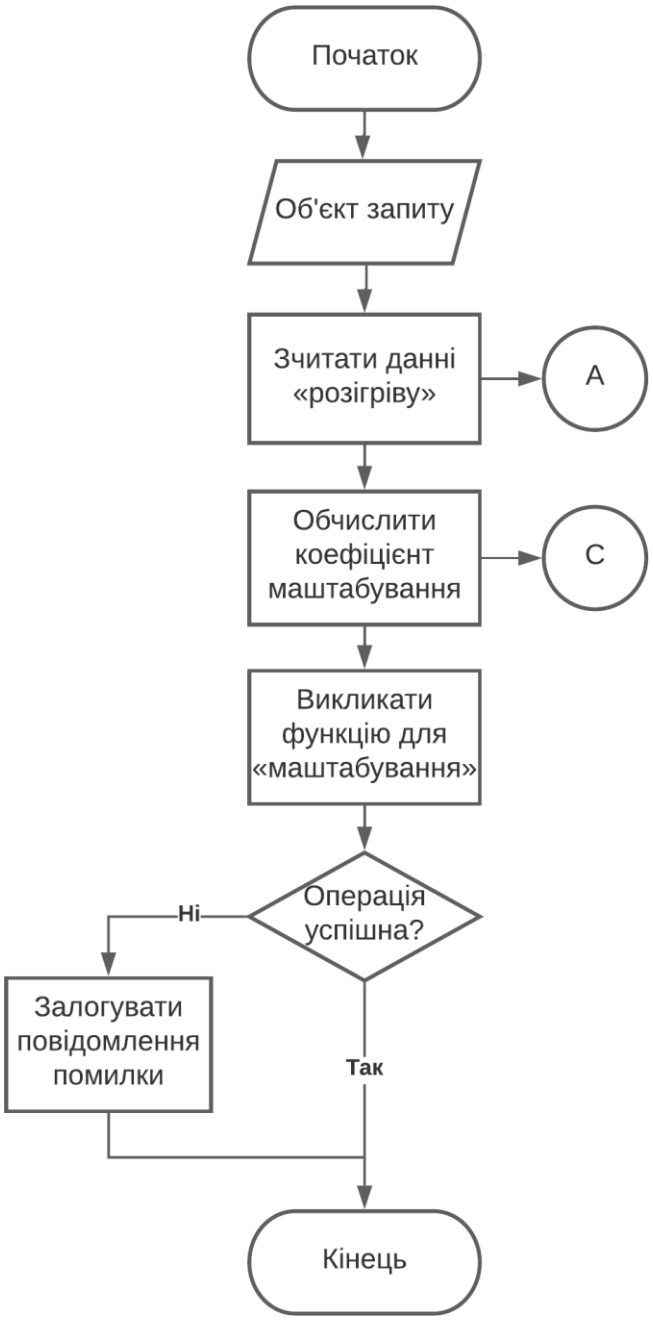

Рис. 3.9 Блок-схема алгоритму програми Lambda функції для «маштабування» тестової системи

Блок-схема алгоритму обчислення коефіцієнта маштабування зображена на рис. 3.10.

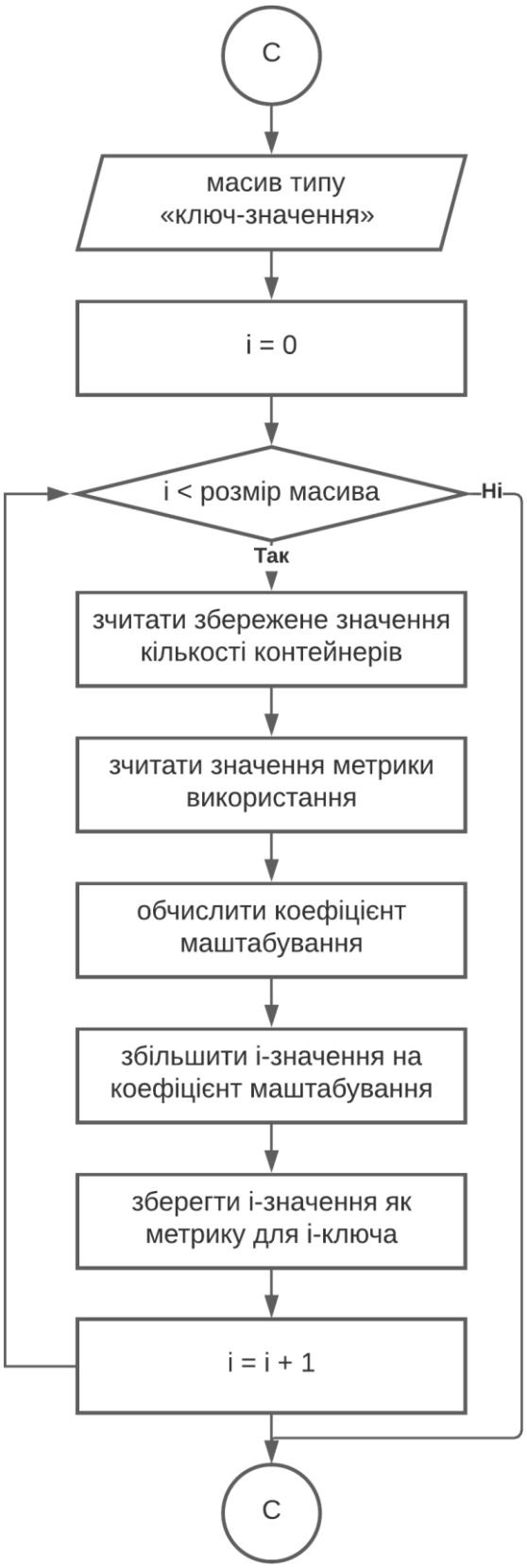

Рис. 3.10 Блок-схема алгоритму процесу обчислення коефіцієнта маштабування Lambda функції тестової системи

Зберігати та зчитувати значення кількості контейнерів M ми будемо за допомогою відповідних запитів до Amazon CloudWatch Metrics.

Так як для кожного екземпляра Lambda функції завжди створюється окремий Log Stream, щоб публікувати туди повідомлення під час виконання програми, зчитувати поточне значення метрики використання ми будемо за допомогою запиту до Amazon CloudWatch Logs, який буде рахувати кількість Log Streams нашої функції у яких були повідомлення про обробку запиту від користувачів за останні N хвилин. Таким чином, дізнавшись кількість екземплярів функції які оброблювали запити від користувачів за останні N хвилин, ми зможемо обчислити використання цієї функції, прирівняши це число до заданої кількості контейнерів M, прийнятого за 100%.

Для розгортання Lambda функцій я використовував інструменти для роботи з AWS які є вседоступними та інтегруються в середовище розробки. Перед розгортанням тестової системи я створив та налаштував свій акаунт, за допомогою інструкцій які доступні на офіційному сайті хмарного провайдера Amazon.

Висновки

1. Поставлено завдання вдосконалити метод активації веб-сервісів на основі безсерверних хмарних обчислень та інтегрувати його до тестової системи.
2. Розроблено програму Lambda функції для «розігріву», яка буде асинхронно викликати функції тестової системи M раз. Розроблено програму Lambda функції для «маштабування», яка дозволяє обчислювати значення асинхронних «пінг» викликів M на основі метрики використання та передавати його як параметр Lambda функції для «розігріву». Блок-схеми алгоритму програм функцій для «маштабування» та «розігріву» зображено на окремих рисунках для простоти читання.
3. Написано код програм Lambda функцій для «маштабування» та «розігріву» з використанням стандартної бібліотеки AWS Lambda SDK та розгорнуто ці Lambda функції в інфраструктурі тестової системи.

РОЗДІЛ 4
ОЦІНКА ЕФЕКТИВНОСТІ ЗАПРОПОНОВАНОГО РІШЕННЯ НА ОСНОВІ НАТУРНОГО МОДЕЛЮВАННЯ

4.1. Аналіз ефективності використання та вимірювання часу відгуку веб-сервісів тестової системи

Мій експеримент для тестової системи моделював від 3 до 20 віртуальних користувачів, посилаючи POST /questions API запити кожної секунди. Для візуалізації результатів експерименту я використовув сервіс AWS X-Ray, який збирав значення часу відгуку Lambda функції PostQuestions. Крім тестових запитів функції PostQuestions, її напряму кожні 5 хвилин викликала Lambda функція для «розігріву», передаючи їй на вхід «фіктивну» подію. Щоб мати бажану кількість підготовлених контейнерів веб-сервісу, вона викликала функцію асинхронно M раз, де за M було прийнято взяти максимальну кількість запитів за секунду, значення якої 20, помножену на середній час виконання функції (250 мс) – тобто 5. Щоб рівномірно розподілити виклики по всім контейнерам, за час «сну» функції PostQuestions було прийнято взяти значення 150 мс. Щоб зберегти максимальну кількість активних екземплярів було прийнято рішення викликати функцію для «розігріву» за розкладом кожні 5 хвилин за допомогою сервісу CloudWatch Scheduled Events.

Інтегрувавши метод активації до тестової системи я запустив тест навантаження для вибраної ділянки веб-додатку. Результати показують середню затримку 175 мс, а p95 – 242 мс і найповільніше виконання 1231 мс (рис. 4.1).

```
All virtual users finished
Summary report @ 13:21:17(+0000) 2020-05-15
  Scenarios launched:  2400
  Scenarios completed: 2400
  Requests completed:  2400
  Mean response/sec: 19.93
  Response time (msec):
    min: 129.9
    max: 1231.9
    median: 174.7
    p95: 242.5
    p99: 591.1
  Scenario counts:
    0: 2400 (100%)
  Codes:
    200: 2400
```

Рис. 4.1 Результати тестування навантаження ділянки веб-додатку Ask Around Me після інтегрування методу активації

В AWS X-Ray графік розподілу часу відгуку показує суттєво покращену продуктивність для маркера p95 (рис. 4.2).

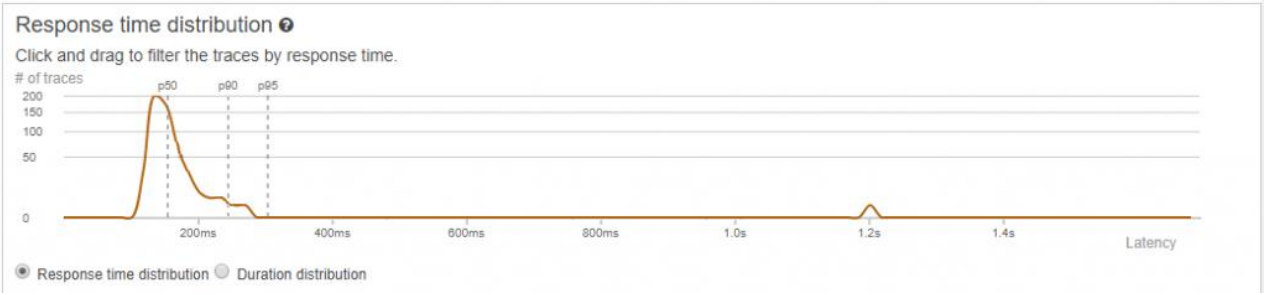

Рис. 4.2 Результати тестування навантаження ділянки веб-додатку Ask Around Me в AWS X-Ray після інтегрування методу активації

Як видно з графіку, після інтегрування методу активації для цієї функції, продуктивність була покращена майже в 4 рази для 95% запитів, а для 1% – все ще спостерігалася затримка через «холодний старт». Через ріст трафіка, фіксованої кількості активних екземплярів функції було замало, що призвело до створення нових середовищ виконання платформою AWS Lambda. Для нашого експерименту цей результат не такий важливий, але якщо це реальний веб-додаток з 400 запитами на секунду під час пікової нагрузки та середнім часом виконання функції в 1 секунду (в 5 разів

повільніше), то затримка в 6 секунд для навіть для 1% буде критичною, а активувати 200 екземплярів функції кожні 5 хвилин щоб колись мати змогу поглинути це навантаження – не ефективно та влетить в копійку власнику веб-додатку. Тому, саме для вирішення цієї проблеми і було запропоновано вдосконалити метод активації веб-сервісів, щоб покращити ефективність використання функцій під час зміни трафіку та зменшити час відгуку більшості запитів.

4.2. Аналіз ефективності використання та часу відгуку веб-сервісів тестової системи після вдосконалення методу активації

Наступний експеримент так само моделював від 3 до 20 віртуальних користувачів, кожен з яких посилає 1 запит на секунду. Функція PostQuestions так само асинхронно викликалась M раз Lambda функцією для «розігріву», яка виконувалась кожні 5 хвилин отримувавши значення M на вхід, але на цей раз вона викликалась не планувальником CloudWatch Scheduled Events, а іншою Lambda функцією для «маштабування», яка це значення M обраховувала. А викликаючи Lambda функцію для «маштабування» за розкладом кожні 5 хвилин ми зберегли цілісність методу активації як і в попередньому експерименті. Результати цього експерименту наведено на рис. 4.3.

```
All virtual users finished
Summary report @ 13:14:35(+0000) 2020-05-19
  Scenarios launched:  2400
  Scenarios completed: 2400
  Requests completed:  2400
  Mean response/sec: 19.17
  Response time (msec):
    min: 136.7
    max: 217.9
    median: 175.9
    p95: 195.5
    p99: 205.1
  Scenario counts:
    0: 2400 (100%)
  Codes:
    200: 2400
```

Рис. 4.3 Результати тестування навантаження ділянки веб-додатку Ask Around Me після вдосконалення методу активації

В AWS X-Ray значення часу відгуку відтепер не розкидані по всій часовій осі, а згруповані близько медіани (рис. 4.4).

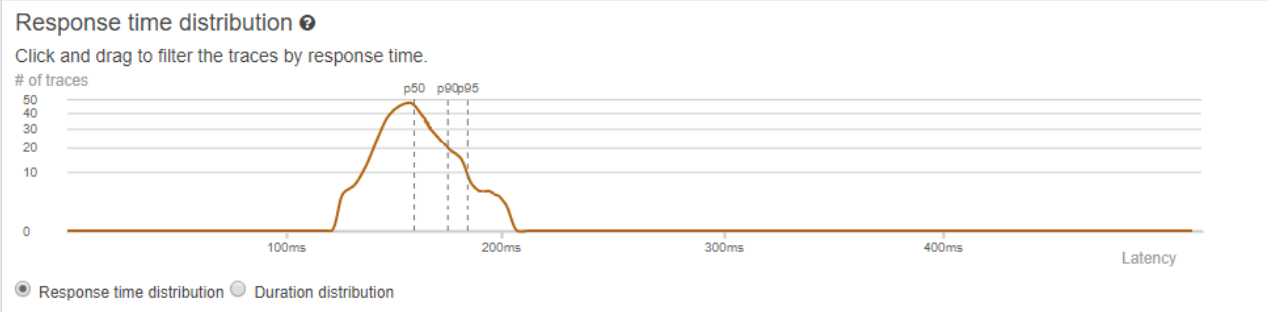

Рис. 4.4 Результати тестування навантаження ділянки веб-додатку Ask Around Me в AWS X-Ray після вдосконалення методу активації

Максимальне значення також значно менше, ніж до вдосконалення методу, що говорить про відсутніть «холодних стартів» взагалі.

Перш за все, ми переконалися, що вдосконалений метод активації веб-сервісів на основі безсерверних хмарних обчислень працює так, як ми його сконфігурували та має меншу кількість «холодних стартів» у системах з непередбачуваним трафіком, а інколи зовсім їх виключає. Відтепер, не потрібно робити надлишкову кількість асинхронних «пінг» запитів, щоб

колись поглинути навантаження під час піку, а достатньо лише додати до методу активації ще одну Lambda функцію, яка буде змінювати їх кількість автоматично, використовуючи мінімум ресурсів.

Висновки

1. Проведено огляд результатів вимірювання часу відгуку веб-сервісів тестової системи протягом. Визначено вплив «холодного старту» на веб-сервіси тестової системи.
2. Проаналізовано результати вимірювання часу відгуку веб-сервісів тестової системи після вдосконалення методу активації. Визначено, які чинники посприяли отриманню таких результатів.
3. Проведено порівняння результатів вимірювання часу відгуку веб-сервісів існуючої системи до та після вдосконалення методу активації веб-сервісів. Визначено, що після вдосконалення методу активації навантаження на веб-сервіси поглинається ефективніше, за рахунок динамічного маштабування. Побудовано графіки розподілу значень часу відгуку функцій тестової системи до та після вдосконалення.

РОЗДІЛ 5
РОЗРОБКА СТАРТАП-ПРОЕКТУ

В цьому розділі буде проведено аналіз стартап проекту «Удосконалений метод активації веб-сервісів на основі безсерверних хмарних обчислень».

Стартап як форма малого ризикового (венчурного) підприємництва впродовж останнього десятиліття набула широкого розповсюдження у світі через зниження бар'єрів входу в ринок (із появою Інтернету як інструменту комунікацій та збуту стало простіше знаходити споживачів та інвесторів, займатись пошуком ресурсів, перетинати кордони між ринками різних країн), і вважається однією із наріжних складових інноваційної економіки, оскільки за рахунок мобільності, гнучкості та великої кількості стартап-проектів загальна маса інноваційних ідей зростає.

5.1 Опис ідеї проекту

Ідея проекту полягає у використанні функції маштабування для автоматичного обчислення коефіцієнта маштабування веб-сервісів та їх активації, що уточнено в таблиці 5.1.

У таблиці 5.1 зображено зміст ідеї та можливі базові потенційні ринки, в межах яких потрібно шукати групи потенційних клієнтів.

Таблиця 5.1.

Опис ідеї стартап проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Використання маштабування для динамічної активації веб-сервісів на основі безсерверних хмарних обчислень.	Веб-додатки побудовані за технологією безсерверних хмарних обчислень	Новий метод покращує ефективність веб-сервісів за рахунок зменшення часу відгуку.
		Запропонований алгоритм покращує використання екземплярів цих веб-сервісів при активації.

Отже, пропонується вдосконалений метод активації веб-сервісів на основі безсерверних хмарних обчислень. Загальною метою запропонованого рішення є автоматичне збільшення кількості екземплярів веб-сервісів при зміні навантаження, щоб уникнути збільшення їх часу відгуку. Отже, він покращує

ефективність веб-сервісів вцілому у порівнянні з методом активації, за рахунок автоматичного обчислення коефіцієнту маштабування цих веб-сервісів. Таким чином, це також вказує на те, що запропонований алгоритм досягає кращих результатів, ніж традиційний метод активації який було взято за основу.

Далі проводимо аналіз потенційних техніко-економічних переваг ідеї порівняно із пропозиціями конкурентів:

— визначаємо перелік техніко-економічних властивостей та характеристик ідеї;

— визначаємо попереднє коло конкурентів (проектів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та проводимо збір інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до визначеного вище переліку;

— проводимо порівняльний аналіз показників: для власної ідеї визначено показники, що мають

— а) гірші значення (W, слабкі);

— б) аналогічні (N, нейтральні) значення;

— в) кращі значення (S, сильні) (табл. 5.2).

Таблиця 5.2

Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів		WW	N	S
		Вдосконалений	Метод активації			
1.	Покращення використання веб-сервісів при зміні навантаження	Більш ефективний	Середній			+
2.	Зменшення час відгуку веб-сервісу	Збільшений	Середній			+
3.	Збільшення операційних витрат	Середній	Середній			+
4.	Ускладнення архітектури системи	+	+		+	

Результати натурного моделювання в тестовій системі підтвердили, що запропонований алгоритм справляється зі зміною навантаження веб-сервісу краще. Отже, ефективність веб-сервісів вцілому покращується, у порівнянні методом активації який було взято за основу, за рахунок динамічної активації екземплярів веб-сервісів тестової системи.

5.2 Технологічний аудит ідеї проекту

В межах даного підрозділу проводимо аудит технології, за допомогою якої можна реалізувати ідею створення проекту.

Визначення технологічної здійсненності ідеї проекту передбачає аналіз складових які вказані в таблиці 5.3.

Таблиця 5.3

Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність техно-логій
1.	Метод активації веб-сервісів на основі безсерверних хмарних обчислень з використанням профілю динамічного маштабування	Статистика	Наяна	Доступна
		Експериментальні дослідження	Наявна	Доступна
		Тестування	Наявна	Доступна
Обрана технологія реалізації ідеї проекту: наявна та доступна на ринку				

Проаналізувавши таблицю можна зробити висновок, що наш проект має достатньо умов для перевірки своєї точності, базисних оцінок, на яких формується проблематика.

5.3 Аналіз ринкових можливостей запуску стартап проекту

Визначимо ринкові можливості, які можна використати під час ринкового впровадження проекту, та ринкові загрози, які можуть перешкодити його реалізації.

Це дозволяє оцінити актуальність нашого проекту.

Спочатку проведемо аналіз попиту: наявність попиту, обсяг, динаміка розвитку ринку (таблиця 5.4).

Таблиця 5.4

Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	2
2	Загальний обсяг продаж, грн/ум.од	5000 грн
3	Динаміка ринку (якісна оцінка)	Зростаюча
4	Наявність обмежень для входу (вказати характер обмежень)	Наявнісь сертифікату відповідності тех.регламенту
5	Середня норма рентабельності в галузі (або по ринку), %	20%

Рентабельність — поняття, що характеризує економічну ефективність виробництва, за якої за рахунок грошової виручки від реалізації продукції (робіт, послуг) повністю відшкодовує витрати на її виробництво й одержується прибуток як головне джерело розширеного відтворення.
З даної таблиці можна зробити висновок, що ринок є привабливим для входження за попереднім оцінюванням.

Цільова аудиторія проекту — компанії які використовують безсерверні обчислення для веб-розробки. Цей ринок достатньо широкий, тому його динаміка є саме зростаючою.

Надалі визначаємо потенційні групи клієнтів, їх характеристики, та формуємо орієнтовний перелік вимог до товару для кожної групи (табл. 5.5).

Таблиця 5.5

Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Ефективність веб-сервісу вцілому	Власники веб-сервісів побудованих за технологією безсерверних обчислень	Вартість проекту.	Зменшений час відгуку веб-сервісу
2	Ефективність використання екземплярів веб-сервісу при зміні навантаження	Власники веб-сервісів які розігріваються за допомогою методу активації	Вартість проекту.	Підвищення ефективності використання екземплярів веб-сервісу

У зв’язку з тим, що аудиторія достатньо широка, викликати довіру новими рішеннями буде не дуже скаладно. Але технологія повинна дійсно підвищувати ефективність використання екземплярів веб-сервісів на основі безсерверних хмарних обчислень. А як показують виміри запропонований алгоритм працює краще, ніж метод на основі якого він був розроблений.

При застосуванні даної технології існують певні загрози. (таблиця 5.6).

Так як прямий споживач — може бути як пересічний власник веб-сервісів, так і інші компанії, узгодження таких змін у архітектурі системи займає багато ресурсів. Проте це не є єдиною проблемою.

Таблиця 5.6

Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Попиту	Вдосконалення може виявитися не настільки потрібним.	Перерахунок вартостей для підтвердження ефективності
2.	Економічна	Зростання інфляції	Пошук можливостей для дешевшого тестування
3.	Конкуренція	Можливо буде розроблений більш ефективний алгоритм	Збільшення перевірок та гарних відгуків
4.	Науково-технічна	Швидкий розвиток платформ безсерверних обчислень хмарних провайдерів	Моніторинг наукових новин та пошук нових шляхів вдосконалення проекту

Ризики існують, тож потрібно мати міцний фундамент у вигляді документів, сертифікатів, які підтверджують усі можливі наміри, результати тестувань та виділення основних переваг цього методу для більшої ефективності використання веб-сервісів на основі безсерверних хмарних обчислень. Але поряд із колом загроз існують і певні можливості (таблиця 5.7).

Таблиця 5.7

Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1.	Конкуренція	Немає аналогів	Збільшення обсягів інтеграції
2.	Економічна	Зменшення операційних витрат	Зниження собівартості
3.	Попиту	Інтеграція зможе створювати перевагу у поєднанні кількох систем одночасно за ціною однієї	Викликання довіри
4.	Природні та екологічні чинники	Підвищення потреби у використанні безсерверних обчислень як більш зеленої технології	Попит
5.	Збуту	Зменшення кола рішень до однієї компанії	Закріплення за собою лідерства у галузі

Деякі загрози можуть слугувати факторами розвитку нових можливостей для проекту. Це звісно спонукає до використання додаткових ресурсів для вирішення цих проблем.

Конкуреція також була як і фактором загрози, так і можливістю показати свої переваги. Для цього був проведений надалі аналіз пропозиції: визначаються загальні риси конкуренції на ринку

Таблиця 5.8

Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Тип конкуренції: олігополія	Невелика кількість фірм на ринку	Підтримка високої якості обслуговування
За рівнем конкурентної боротьби: національний	Багато систем використовують старі інструменти підвищення ефективності	Ведучи конкуренцію на національному рівні, компанії необхідно прикласти належні зусилля для охоплення всього національного ринку
За галузевою ознакою: внутрішньогалузева	Стосується тільки галузі безсерверних хмарних обчислень	Необхідно зосередити зусилля на пошуку конкурентних переваг, які дозволять компанії займати стійкі конкурентні позиції на даному ринку
Конкуренція за видами товарів: товарно-родова	Між іншими потенційними рішеннями	Покращувати рекламу
За характером конкурентних переваг: цінова	Споживач звертає увагу на те, скільки коштуватиме інтеграція нашого проекту у його продукт	Пошук підрядників, які б виконувати роботі процеси за нижчу ціну
За інтенсивністю: марочна	Один відомий продукт бренду може принести продажі інших продуктів. Тож з’явиться сенс покращувати актуальні продукти.	Набір усіх необхідних документів та даних для легкої та вдалої інтеграції

Аналіз підтвержує, що навіть при свою специфіку, наш проект потребує значних зусиль для того, щоб увійти у ринок, зафіксуватися та пропонувати свої можливості своїм клієнтам. І це як раз той випадок, коли вартість впливає на прийняття рішення.

Після аналізу конкуренції проведемо більш детальний аналіз умов конкуренції в галузі.

Таблиця 5.9

Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенці-йні конкурен-ти	Постачальники	Клієнти	Товари-замін-ники
	Метод активації	Можливі нові методи	Робоча сила, елементи у системі вдосконалення	Ціноутво-рення	Неякісні замін-ники
Висновки	Немає високої конкуренції, оскільки розроблений метод перевершує вже існуючі рішення, але є так звана «перевага першопрохідців»	Нові рішення потенційно можуть мати перевагу над розроблени м нами.	-	Клієнти диктують основні умови на ринку	-

Після всіх аналізів визначається та обґрунтовуєтеся перелік факторів конкурентоспроможності.

Таблиця 5.10

Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Ціна	Ціна інтеграція впливає на прийняття рішення. А наша ціна вигідніше, ніж у аналогів.
2	Актуальність	Вдосконалюється метод, проте має бути важлива основа, яка підтверджує актуальність. І вона доведена нашим проектом.
3	Попит	Наука розвивається, як і інформаційні системи. І технологія не може бути несучасною.
4	Енергоефективність	Наш метод є найбільш ефективним на ринку.
5	Інноваційність	Робить українську науку на рівні с іншими країнами.

Фінальним етапом ринкового аналізу можливостей впровадження проекту є складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) (табл. 5.12) на основі виділених ринкових загроз та можливостей, та сильних і

слабких сторін (табл. 5.11). Перелік ринкових загроз та ринкових можливостей складається на основі аналізу факторів загроз та факторів можливостей маркетингового середовища. Ринкові загрози та ринкові можливості є наслідками (прогнозованими результатами) впливу факторів, і, на відміну від них, ще не є реалізованими на ринку та мають певну ймовірність здійснення.

Таблиця 5.11

Порівняльний аналіз сильних та слабких сторін

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів						
			–3	–2	–1	0	+1	+2	+3
1	Ефективність відгуку	16					+		
2	Використання ресурсів	18						+	

З таблиць 5.10 та 5.11 бачимо, що фактори конкурентоспроможності суттєві та мають великий позитивний внесок при впровадженні нового програмного забезпечення для розрахунку концентрації пилу. Основною перевагою та головним досягненням є висока якість продукту та технічна підтримка на протязі всього терміну його використання споживачем.

Таблиця 5.12

SWOT- аналіз стартап-проекту

Сильні сторони: 3 Невелика конкуренція; 4 Інноваційність; 5 Вартість.	Слабкі сторони: 1) Відсутність довіри; 2) Велика витрата ресурсів до самих продажів на рекламу
Можливості: 1. Збільшення продаж; 2. Отримання державних замовлень на отримання послуг; 3. Розширення ринку за рахунок іноземних замовників; 4. Отримання тендерів на послуги.	Загрози: — Цінова конкуренція в зв'язку з появою нових гравців на ринку. — Різка зміна курсу гривні може привести до зменшення попиту, особливо з боку малих фірм.

Це знову підтверджує, що навіть незважаючи на свою специфіку, наш проект потребує значних зусиль для того, щоб увійти у ринок, зафіксуватися та пропонувати свої можливості своїм клієнтам.

На основі SWOT-аналізу розробляємо альтернативи ринкової.

Таблиця 5.13

Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Стратегія нейтралізації ринкових загроз сильними сторонами стартапу	70%	3 місяці
2	Стратегія компенсації слабких сторін стартапу наявними ринковими можливостями	70%	3 місяці
3	Стратегія виходу з ринку	80%	6 місяців

З зазначених альтернатив обираємо стратегію компенсації слабких сторін стартапу наявними ринковими можливостями.

5.4 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів.

Таблиця 5.14

Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Власники веб-додатків	Так	Середній	Середня	Складна
2	Веб-розробники	Так	Високий	Висока	Складна
3	Оператори інфраструктур	Так, але складніше	Середній	Низька	Складна
Які цільові групи обрано: Під час аналізу потенційних груп споживачів було прийнято рішення що компанія буде працювати із безсерверними хмарними технологіями					

За результатами аналізу потенційних груп споживачів ми обрали цільові групи, яким найбільш необхідний наша розробка. Адже тільки вони можуть інтегрувати його у свої продукти, тим самим вдосконалюючи їх, тестувати, робити висновки та використовувати у комерційній діяльності.

Для роботи в обраному сегменті ринку необхідно сформувати базову стратегію розвитку.

Таблиця 5.15

Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку*
1	Підсилення сильних сторін стартапу за рахунок ринкових можливостей	Перемовини з компаніями, які представляють цільові групи потенційних клієнтів	Видокремлення переваг цього способу у грошовому еквіваленті для майбутніх споживачів.	Стратегія підкріплення своїх переваг

Наступним кроком є вибір стратегії конкурентної поведінки (табл. 5.16).

Таблиця 5.16

Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Ні	Забирати існуючих	Ні	Стратегія підкріплення своїх переваг

На основі вимог споживачів з обраного сегменту до постачальника і продукту, а також в залежності від стратегії розвитку та стратегії конкурентної поведінки розробляємо стратегію позиціювання яка визначається у формування ринкової позиції, за яким споживачі мають ідентифікувати проект.

Таблиця 5.17

Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Цілкова підтримка на етапі інтеграції	Відкритість до вирішення питань	Обізнаність свого продукту, допомога в інтеграції. Формування лояльності і прихильності споживачів, підтримка вхідних бар'єрів.	Якість. Ефективність відгуку. Використання екземплярів веб-сервісу.

Результатом даного підрозділу є система рішень щодо ринкової поведінки компанії, вона визначає в якому напрямі буде працювати компанія на ринку

5.5 Розроблення маркетингової програми стартап-проекту

Під час розроблення маркетингової програми першим кроком є розробка маркетингової концепції товару, який отримає споживач. У таблиці 5.18 підсумовуємо результати аналізу конкурентоспроможності товару.

Таблиця 5.18

Визначення ключових переваг концепції товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Конкурентоспроможності	Унікальність	Немає анонсованих вдосконалень

Це основна причина споживачів придбати наш проект — стати унікальними на ринку.

Таблиця 5.19

Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Довго вагаються для прийняття рішень	-	Унікальність	Донести, що завдяки нашому проекту буде прибуток	Унікальність

Висновки

Узагальнюючи проведений аналіз стартап проекту можна зробити висновок, що проект «Удосконалений метод активації веб-сервісів на основі безсерверних хмарних обчислень» є реальним, проте має багато ризиків. Вдалося прорахувати його можливості на ринку та загрози. Зараз на нашому ринку немає анонсованих аналогів подібного способу, проте можливо, що згодом вони з’являться. Проте це може створити ряд перепон, як технічних, бюрократичних, так і фінансових. Тож завдання інтегрувати розроблений метод у продукти наших потенційних клієнтів є реальним, але має мати щось більше, ніж просто обіцяючі аргументи. Це мають бути сертифікати, статистичні дані, багато досліджень щодо необхідності цього способу та доведення, що спосіб не суперечитиме існуючим діям бездротових сенсорних систем. Адже саме це є основою у вдосконаленні веб-сервісів побудованих за технологією безсерверних хмарних обчислень.

Також можна зробити висновок, що значну роль відіграватиме вартість інтеграції. Це те, що у першу чергу впливатиме на рішення власника веб-додатку, адже кінцева вартість його продукту має бути конкурентоспроможною.

Так як галузь потенційно достатньо широка в Україні, наш проект у теорії матиме попит серед наших розробників веб-сервісів які побудовані за технологією безсерверних хмарних обчисленьц.

Наступний висновок — так як інші виробники ще не анонсували подібних вдосконалень, у проекту є шанси стати лідером у своїй області. А продукт, який інтегрує його у себе — монополістом.

ЗАГАЛЬНІ ВИСНОВКИ ПО РОБОТІ

1. Проаналізовано проблеми ефективності використання веб-сервісів на основі безсерверних хмарних обчислень. Визначено, що головними проблемами таких веб-сервісів є «холодний старт» та непередбачуваність трафіку реальних систем.
2. Проведено огляд існуючих методів вирішення проблем ефективності використання веб-сервісів на основі безсерверних хмарних обчислень та поставлено завдання вдосконалити метод активації для покращення часу відгуку цих веб-сервісів та ефективності використання їх екземплярів.
3. Удосконалено метод активації веб-сервісів на основі безсерверних хмарних обчислень за рахунок автоматичного обчислення коефіцієнта маштабування екземплярів цих веб-сервісів. При зміні навантаження на веб-сервіс, кількість його екземплярів змінюється автоматично, при цьому, тримаючи час відгук веб-сервісу постійно низьким.
4. Проведено натурне моделювання запропонованого рішення, що підтвердило його працездатність. Графік розподілу часу відгуку показав покращення ефективності використання веб-сервісу за рахунок зменшення використання його активних екземплярів при зміні трафіку та покращення ефективності вцілому (зменшення часу відгуку) для 99% запитів.
5. Розроблено стартап-проект для запропонованого рішення на снові дослідження ринку та результатів натурного моделювання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Черешня В. Р. Підвищення ефективності веб-сервісів на основі хмарних безсерверних обчислень / В. Р. Черешня, В. В. Курдеча // Дипломна робота на здобуття ступеня бакалавра з напряму підготовки 6.050903 «Телекомунікації». – 2019. – С. 10–57.

2. Черешня В. Р. Вдосконалений метод активації веб-сервісів на основі безсерверних хмарних обчислень / В. Р. Черешня, В. В. Курдеча, Скулиш М.А. // Збірник матеріалів тринадцятої науково-практичної конференції "Пріоритетні напрямки розвитку телекомунікаційних систем та мереж спеціального призначення. Застосування підрозділів, комплексів, засобів зв'язку, автоматизації та кібербезпеки в операції Об'єднаних сил". – 2020. – С. 289–290.

3. Черешня В. Р. Оцінка ефективності веб-сервісів за технологією безсерверних хмарних обчислень / В. Р. Черешня, В. В. Курдеча. // Матеріали міжнародної науково-технічної конференції "ПЕРСПЕКТИВИ ТЕЛЕКОМУНІКАЦІЙ". – 2019. – №2663. – С. 239–241.

4. F. Munz. Serverless Architectures. AWS Lambda and Fn Project / F. Munz. // presented at Devoxx, Casablanca, Morocco, 2017.

5. Şamdan E. Dealing with cold starts in AWS Lambda [Електронний ресурс] / Emrah Şamdan. – 2018. – Режим доступу до ресурсу: https://medium.com/thundra/dealing-with-cold-starts-in-aws-lambda-a5e3aa8f532.

6. Become a Serverless Black Belt: Optimizing Your Serverless Applications – SRV401 / presented at AWS re:Invent, Las Vegas, USA, 2017.

7. Artillery - a modern load testing toolkit [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://artillery.io.

8. Windisch E. Understanding AWS Lambda Coldstarts [Електронний ресурс] / Erica Windisch – Режим доступу до ресурсу: https://www.iopipe.com/2016/09/understanding-aws-lambda-coldstarts.

9. Cui Y. How long does AWS Lambda keep your idle functions around before a cold start? [Електронний ресурс] / Yan Cui – Режим доступу до ресурсу: https://read.acloud.guru/how-long-does-aws-lambda-keep-your-idle-functions-around-before-a-cold-start-bf715d3b810.

10. Roberts M. Serverless Architectures [Електронний ресурс] / Mike Roberts. – 2018. – Режим доступу до ресурсу: https://martinfowler.com/articles/serverless.html.

11. Zappa - Serverless Python Web Services [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.zappa.io/.

12. AWS Lambda – Serverless Compute - Amazon Web Services [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: https://aws.amazon.com/lambda/.

13. Wittig A. Amazon Web Services in Action / A. Wittig, M. Wittig., 2018. – 528 с. – (Manning Publications). – (2 edition; 1617295116).

14. Mińkowski P. Serverless on AWS Lambda [Електронний ресурс] / Piotr Mińkowski. – 2017. – Режим доступу до ресурсу: https://piotrminkowski.wordpress.com/2017/06/23/serverless-on-aws-lambda/.

15. Mińkowski P. Serverless on AWS with DynamoDB, SNS and CloudWatch [Електронний ресурс] / Piotr Mińkowski. – 2017. – Режим доступу до ресурсу: https://piotrminkowski.wordpress.com/2017/07/03/serverless-on-aws-with-dynamodb-sns-and-cloudwatch/.

16. Thundra Lambda Warmup [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://github.com/thundra-io/thundra-lambda-warmup.

17. Poccia D. AWS Lambda in Action: Event-driven Serverless Applications / Danilo Poccia., 2016. – 384 с. – (Manning Publications). – (2 edition; 2147483647)

18. Daly J. 15 Key Takeaways from the Serverless Talk at AWS Startup Day [Електронний ресурс] / Jeremy Daly. – 2018. – Режим доступу до ресурсу: https://www.jeremydaly.com/15-key-takeaways-from-the-serverless-talk-at-aws-startup-day/.

19. AWS Toolkit for Eclipse [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://aws.amazon.com/eclipse/.

20. Sbarski P. Serverless Architectures on AWS / P. Sbarski, A. Nair., 2019. – 500 с. – (Manning Publications). – (2 edition; 1617295426).

21. Roy S. Maven plugin to facilitate AWS Lambda deployments as part of your maven build process [Електронний ресурс] / Sean Roy. – 2019. – Режим доступу до ресурсу: https://github.com/SeanRoy/lambda-maven-plugin.

22. Gurturk C. Building Serverless Architectures / Cagatay Gurturk., 2017. – 242 с. – (Packt Publishing). – (1 edition; 2147483647).